PlayTales® Builder™ Manual

PlayTales® Builder™ User Manual

Carlos Llera Martín, Kate Shoaf, Charlie Bloxham

19/02/2014

Content

Presentation 5

What is PlayTales® Builder™? 6

More about PlayTales® and PlayTales® Builder™? 7

Quick Guide 9

Getting and installing PlayTales® Builder™ 10

The workspace 11

Create a “static” story 12

Create an animated and interactive story 21

Create a “PlayTales” story 31

Test the project on a real device 32

Publish your project 35

Before using the program 37

What does it take to make a book in PlayTales®? 38

What is the structure of a book in PlayTales ® Builder ™? 39

Current Limitations of PlayTales ® Builder ™ 41

Basic Concepts 43

Create a project 44

Add resources to your project 46

Edit Project Properties 48

Create a scene and edit its properties 51

Create Actor and edit its properties 53

Launching PlayTales® Simulator 58

Animations 61

How do animations work in PlayTales® Builder™? 62

Animation Events 62

Animation States 65

Animation Structure 66

Actions 68

TO and BY Modes 68

Easing 69
Interval Actions 72
Instant Actions 75
Sound Actions 80
Custom Actions 81
Create an animation 82
Edit an animation 84
Test your project on a real device 86
Deploy the project from PlayTales® Builder™ 87
Receive project in PlayTales® Builder™ Viewer 89
Deploy a project from a Mac 91
Install Homebrew 92
Install FFmpeg 92
Install ImageMagick 93
Publish your project 95
Post your project from PlayTales® Builder™ 96
Advanced Concepts 99

Presentation

In this section we will briefly introduce **PlayTales® Builder™** and we will identify it with the other **PlayTales®** applications. If you don't want to wait anymore and use the program now, go directly to the *Quick Guide* section.

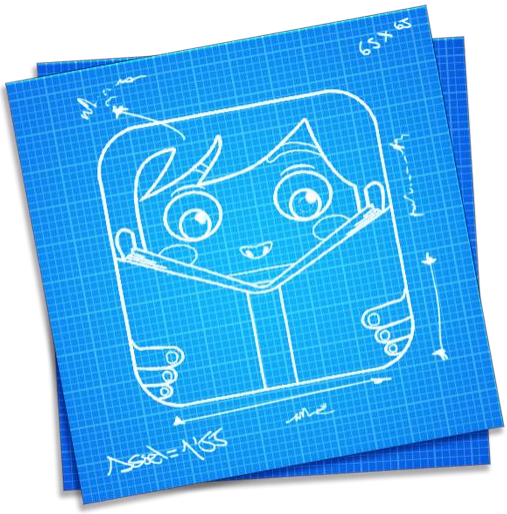

Figure 1 – PlayTales Builder Icon

What is PlayTales® Builder™?

Have you ever wanted to publish a digital book? By using PlayTales® Builder™ you'll not only be able to publish your own digital e-book, but you can add tons of interactive features to your story and sell your book through the top international mobile platforms.

PlayTales® Builder™ is an application designed by Genera Interactive® that enables users to create and publish interactive stories. These digital stories can be sold through PlayTales® Bookstore applications as well as other popular mobile application platforms such as the **Apple Store** or **Google Play**.

One of the great things about PlayTales® Builder™ is the fact that you don't need any prior programming knowledge to create awesome interactive stories! It's easy to use and we've prepared various tools and tutorials to help you get the most out of your Builder experience.

Figure 2 - PlayTales® Builder™ is available in many platforms and devices

PlayTales® Builder™ is available for **PC and Mac**, it's **free**, and you can download it from the **official website**.

For more information about PlayTales® Builder™, please visit the application's official website: www.playtalesbuilder.com

More about PlayTales® and PlayTales® Builder™?

PlayTales® Builder™ is part of the PlayTales® project, which aims to provide young readers with an extensive library of books where they can read, learn, and play.

PlayTales® produces award winning apps featuring unlimited interactive books for kids. PlayTales® interactive books are designed to inspire, educate, and entertain readers ages 1 to 8. The PlayTales® team consists of over 100 individuals dedicated to the production of wholesome and fun mobile content for children.

Figure 3 – The PlayTales® team

PlayTales® is a wholly-owned subsidiary of Genera Interactive, an internationally based multi-platform mobile entertainment and utilities provider with presence in USA, UK, Spain, Romania, and China. We've combined our knowledge and experience to help create the cutting-edge technology featured in Builder and we love to see users like you using the program to create fun and interactive eBooks for kids.

Quick Guide

This section provides a quick guide for you to follow as you begin using the program. The Quick Guide features step-by-step instructions and resource tools that will help make using PlayTales® Builder™ an easy and productive experience.

Figure 4 – *The Three Little Pigs*, from Pablo Gastalver

Getting and installing PlayTales® Builder™

1. Download the latest version of the program from the official website: www.playtalesbuilder.com
2. Run the installer and follow the wizard to install the application on your computer.
    a. The installer includes everything needed to run PlayTales® Builder™ on your computer, but if you have problems running it, make sure you've installed the latest free version of **Adobe AIR**: www.get.adobe.com/air
    b. If you already have an older version of PlayTales® Builder™ and the installer displays an error during installation, uninstall the previous version you have (don't worry, your projects will not be deleted) and continue with the download of the new version.

Now you're ready to start using the program!

Figure 5 – PlayTales Builder

The workspace

We'll spend most of our time working on projects on the Workspace screen.

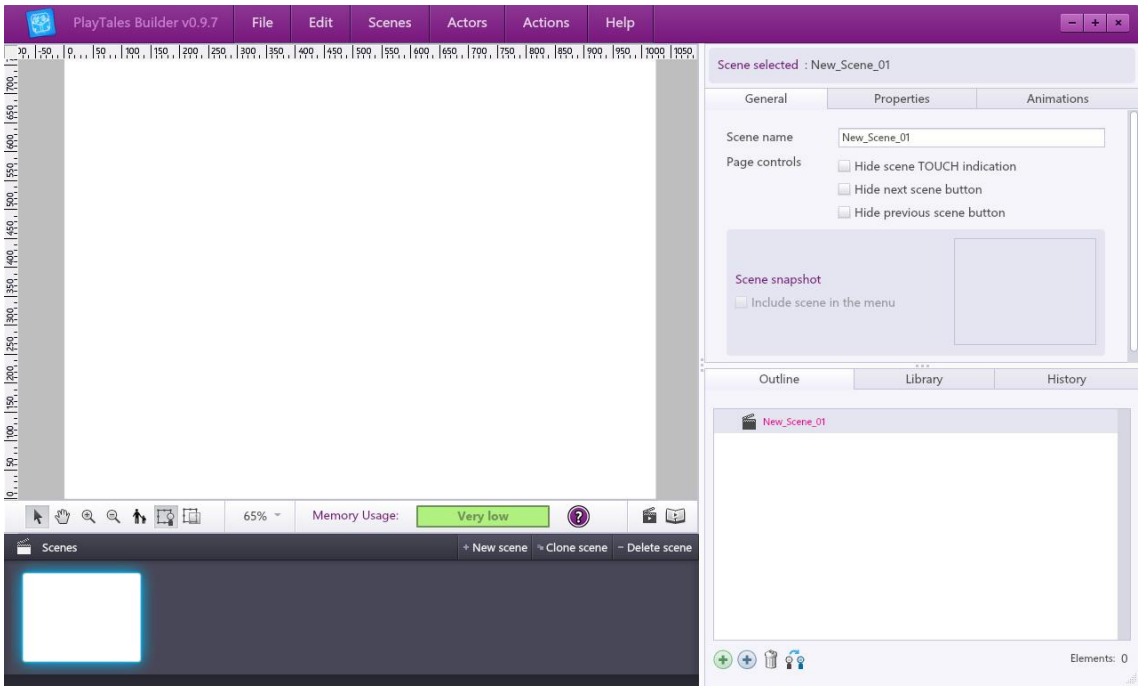

Figure 6 – PlayTales Builder Workspace

The Workspace features the following parts.

1. **Toolbar**: displays all the options available in the program.
2. **Scene Canvas**: it is a WYSIWYG editor of the current scene.
3. **List of scenes**: the scenes that make up the project.
4. **Selected Item Properties**: editable forms.
5. **Outline, Library, and History**: three tabs that show a collection of objects in the scene, project resources, and the respective history of operations.

Create a "static" story

First we'll quickly create multiple scenes of a story without animations, sound, and interactivity... just like a traditional paper book. The resources used in the examples are from **Moon Secrets**, a book originally designed by Emilio Villalba.

You can download the necessary resources from this link:

http://www.playtalesbuilder.com/resources

Follow the steps as they are listed below:

1. Create a new book by clicking on the *new project button*, or by selecting *File* > *New* and fill out the form that appears on the screen.

Figure 7 – New Project button.

Figure 8 – New Project wizard

2. When creating a new project, a first blank scene is added to the list of scenes in the bottom of the program. If this blank scene does not appear, click the *New Scene* button or in the *Scene* menu select *New Scene* to create one.

Figure 9 – New Scene button

3. In the *Library* tab (on the bottom, right-hand side of the screen) you can access the image resources that we will use for the project by clicking on the ***(+) button*** and select the images you want from the System Explorer.

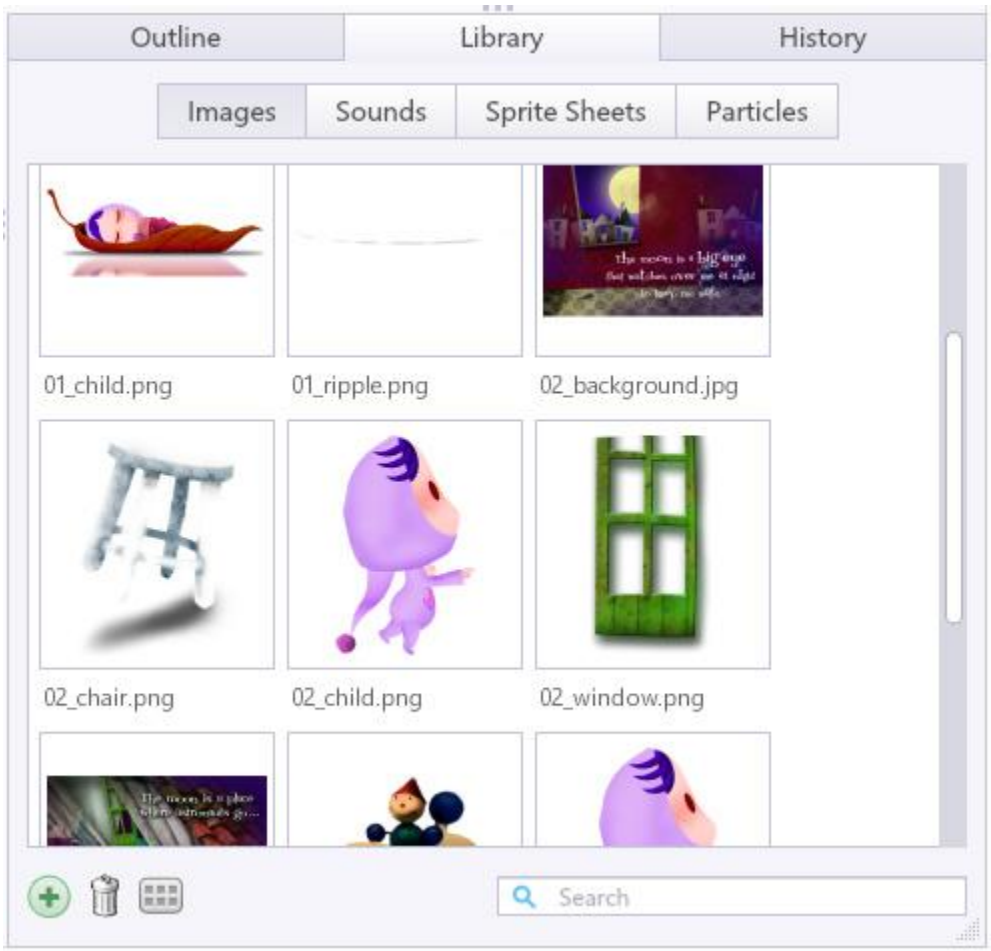

Figure 10 – Library Images

You can add all the resources at once, or gradually as needed. OK, now, we'll add all the files in the *images* folder to the ZIP folder from downloaded resources.

NOTE: It's good to organize your resources by putting names to indicate what scene or at what point you want to use them. I normally add a prefix to the number of the scene where they will be used or "00_" if they will be featured in several scenes.
NOTE: You can also drag image files from Windows Explorer (Windows) or Finder (Mac) to the folder *assets / images* of the project, and they will automatically be available in Builder.

4. To compose the first scene of our project we will add the *Actor* images (which are prefixed with "01_") that belong in that scene. This can be done in two ways:
    a. By dragging images directly from the *Library* onto the canvas of the scene. In this case a new *actor* will be created in the *Outline* tab with the texture and position you selected.
    b. Pressing the ***(+) button*** in the *Outline* tab or via the *Actors* menu> *New Actor*, you will then need to select the image you want to use as the texture.

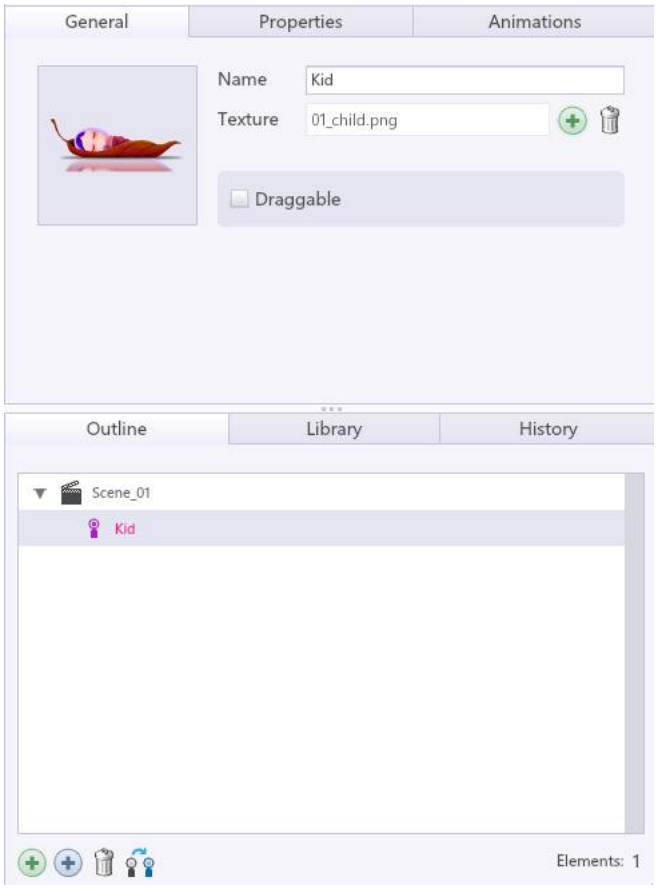

Figure 11 – Create an actor

NOTE: We recommend you always assign a name to the actors and scenes that are created so that you can easily identify them. You can use the *General* tab to do this.

Add the next actors to the first scene you created. You can modify the following values in the *General* and *Properties* tabs.

a. Name: **Background**
   - Texture: 01_background.jpg
   - Position: X = 512, Y = 384
   - Z-Index: -100

b. Name: **Child**
   - Texture: 01_child.png
   - Position: X = 335, Y = 125
   - Z-Index: 100

c. Name: **Ripple_1**
   - Texture: 01_ripple.png
   - Position: X = 330, Y = 70
   - Z-Index: 10

d. Name: **Ripple_2**
   - Texture: 01_ripple.png
   - Position: X = 330, Y = 70
   - Scale: X = 1.5, Y = 1.5
   - Z-Index: 20

For more information on the properties of the scenes, actors and other elements, see the *Basic Concepts* section.

NOTE: To locate resources easily in the library, we can use the search tool located at the bottom right-hand side of the page. Since I have used prefixes in the names, I can use "01_" to find all the resources of the first scene.

Once you've added the actors, the scene should look like this:

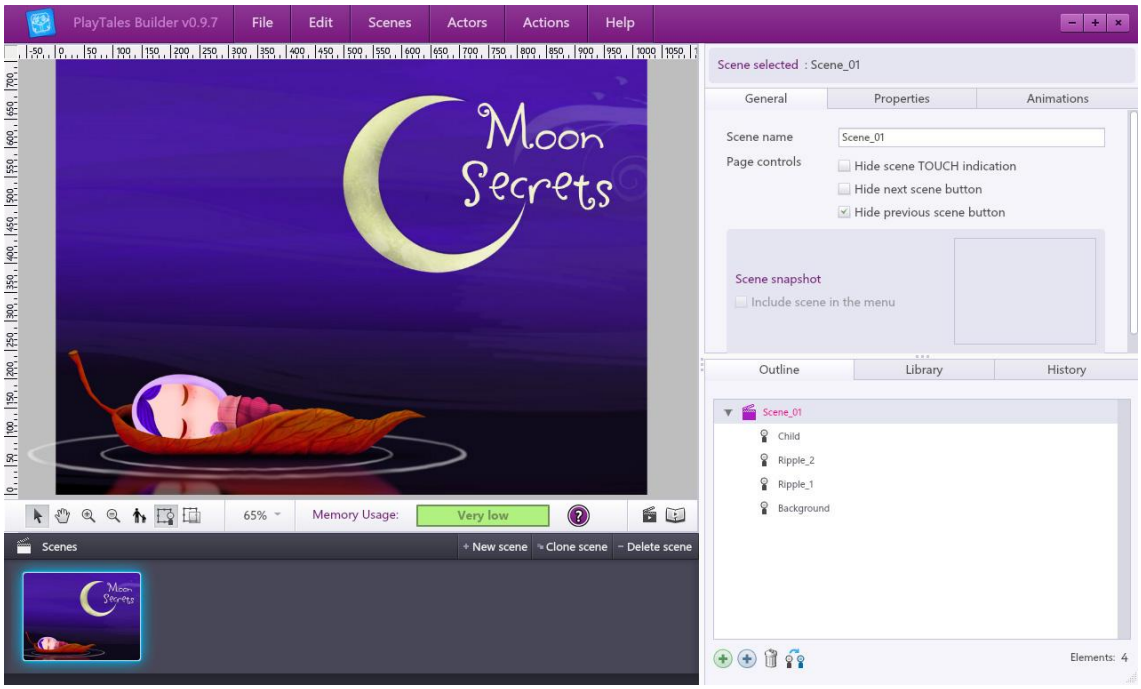

Figure 12 – First scene of *Moon Secrets*

5. Next, we'll create a new scene and compose it in the same way with the following *actors*:
    a. Name: **Background**
        - Texture: 02_background.jpg
        - Position: X = 512, Y = 384
        - Z-Index: -100
    b. Name: **Chair**
        - Texture: 02_chair.png
        - Position: X = 200, Y = 145
        - Z-Index: 50
    c. Name: **Child**
        - Texture: 02_child.png
        - Position: X = 215, Y = 395
        - Z-Index: 100
    d. Name: **Window**
        - Texture: 02_window.png
        - Position: X = 565, Y = 545
        - Z-Index: 200

It should look like the following example:

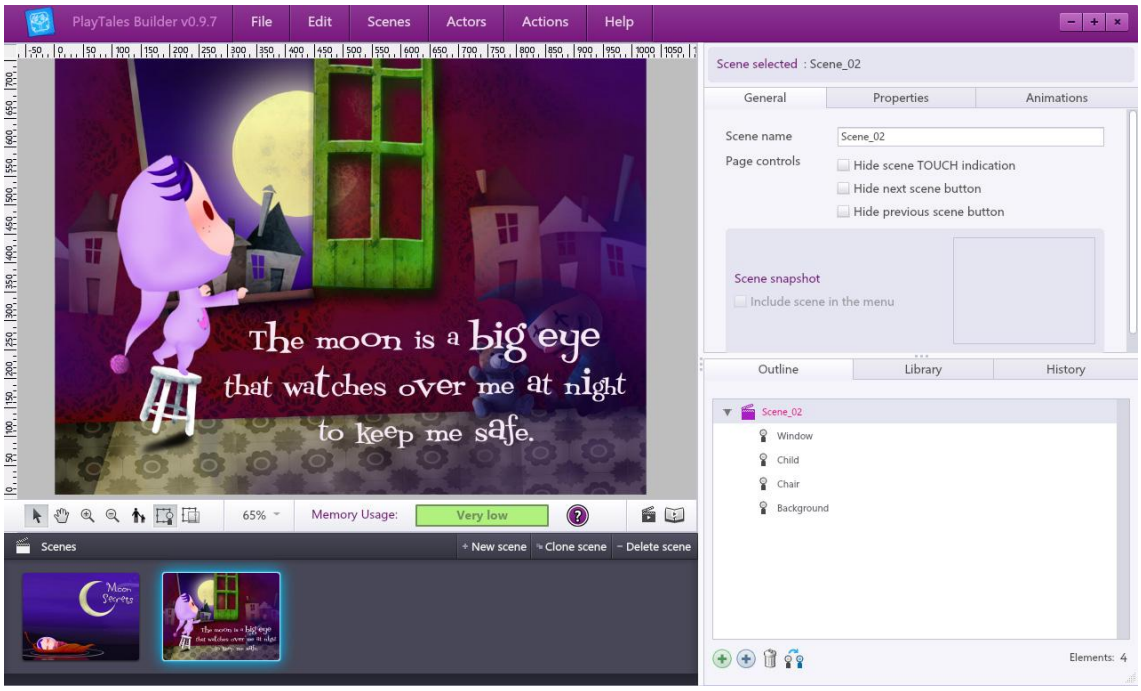

Figure 13 – Second scene of *Moon Secrets*

6. Let's build the third scene like this:
    a. Name: **Background**
        - Texture: 03_background.jpg
        - Position: X = 512, Y = 384
        - Z-Index: -100
    b. Name: **Telescope**
        - Texture: 03_telescope.png
        - Position: X = 775, Y = 315
        - Rotation: 40
        - Z-Index: 10
    c. Name: **Child**
        - Texture: 03_child.png
        - Position: X = 465, Y = 300
        - Z-Index: 5
    d. Name: **Toy**
        - Texture: 03_toy.png
        - Position: X = 160, Y = 295
        - Z-Index: 20
    e. Name: **Car**
        - Texture: 03_car.png

- Position: X = 220, Y = 170
- Z-Index: 30

f. Name: **Plane**

- Texture: 03_plane.png
- Position: X = 160, Y = 650
- Z-Index: 40

And it should look like this:

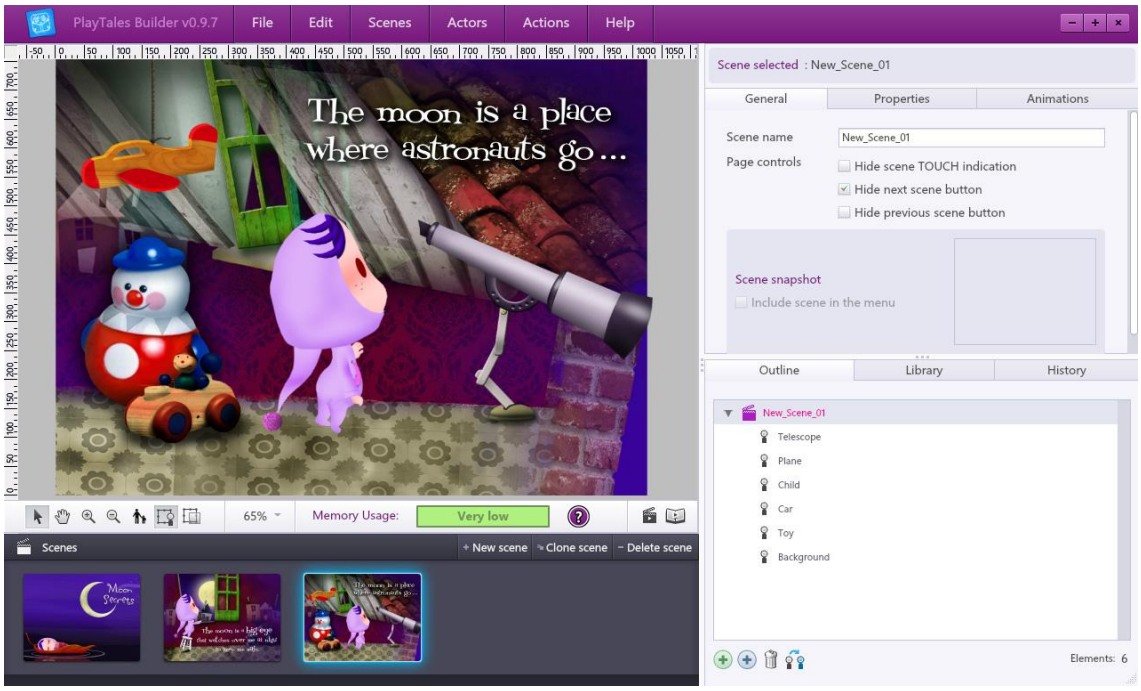

Figure 14 -Third scene of *Moon Secrets*

As we continue constructing scenes, it is essential to see an accurate preview of our work. PlayTales® Builder™ incorporates a simulator that very accurately depicts how your work will appear on a mobile device.

To launch the simulator, you can press the button located at the bottom-right side of the scene canvas or select *Edit > Simulate to tablet* or *Simulate to phone*, and a preview of the scene we have selected is displayed, appearing as it would on a tablet or a phone.

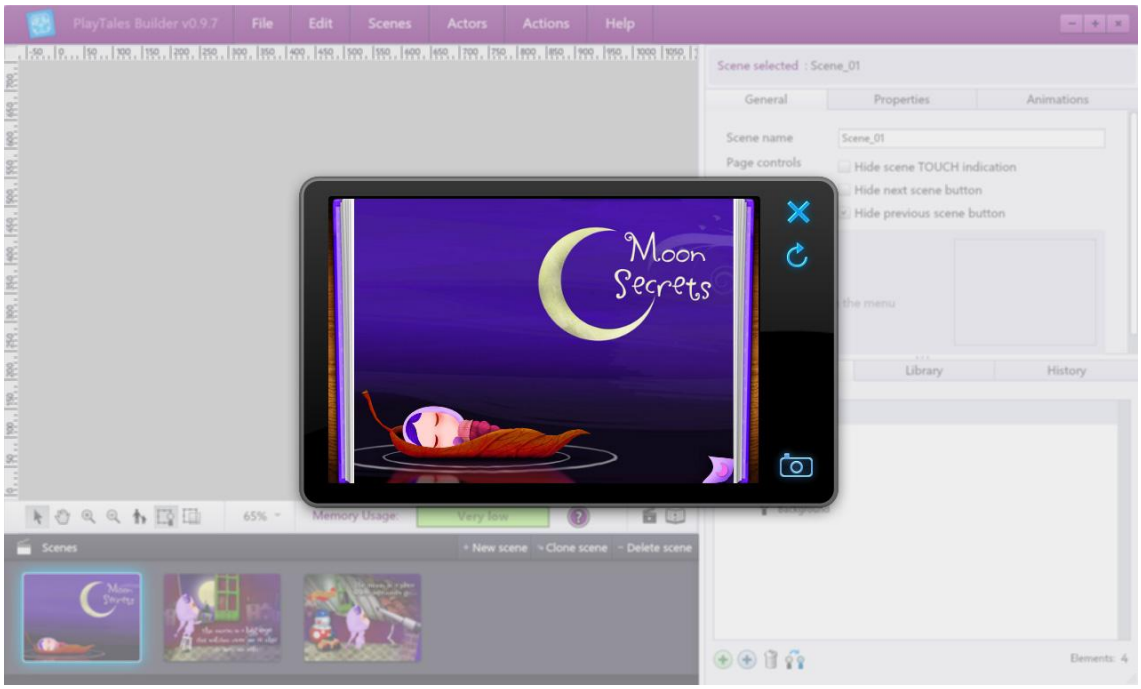

Figure 15 - Simulating *Moon Secrets* to phone

Once you launch the simulator, you can't automatically view other scenes. A quick and easy way to be able to navigate between scenes is to add a "previous page" and "next page" option within each scene. Check out the following instructions to see how this can be done.

7. Open the project properties by clicking the *Edit* > *Edit project properties* menu.
8. In the top right side of the page you will see a form with information about the project properties. Edit the following items in the *Properties* tab:
    a. **Previous Scene** button: 00_previous.png
    b. **Next Scene** button: 00_next.png

Re-launch the simulator, now the navigation buttons should appear so that you can go back and forth between scenes.

Now we have the option to go back to the "previous page" and forward to the "next page" from any page. However, if you don't want people to be able to go from the last page directly to the first page (or the first page directly to the last page) you can open the scene properties form by selecting the scene from the scene list on the bottom of the screen and follow these instructions:

9. For scenes 1 and 3, in the *General* tab, tick the buttons indicating *Hide previous scene* and *Hide next scene*.

That's it! With these well organized and prepared resources we can build a basic story within minutes! With just a little more effort we can bring our characters to life by following the next example.

Create an animated and interactive story

A major advantage of modern mobile devices is that we can add sound, movement, and interaction to the pages of our story. Thanks to the tools of PlayTales® Builder™ we can incorporate these interactive features, modify the graphics of the scene, and give life to our characters in the story quickly and easily.

In this section, we will create some simple animations from scenes made in the previous example. You can download the previous project we just finished and the new resources you'll need from the following address:

http://www.playtalesbuilder.com/resources

Scene 1:

In this scene we will see how we can automatically animate our actors as soon as the scene begins.

1. To animate the *Kid* actor along with the waves in the water, we will make both waves *Ripple_1* and *Ripple_2* children of *Kid* actor. By doing this the graphic properties of the waves depend on their parent actor, *Kid*. For more information on parents and children, please see the *Advanced Concepts* section.
    a. Cut the actor *Ripple_1* by selecting the actor in the scene and pressing **Ctrl + X** on Windows or **Command + X** on Mac or via the *Edit* > *Cut* menu.
    b. Select the *Kid* actor, and **press Ctrl + V** (or **Command + V**) or *Edit* > *Paste* to paste the cut *Ripple_1* as the child of the *Kid* actor.
    c. Do the same for the actor *Ripple_2*.
    d. To make sure the child actors' textures are not overlapping their parent, we need to change the Z-Index property of both to a negative number: -10 to -20 for example.
    e. Reposition the pasted child actors if necessary.

The outline of the scene should look like this:

Figure 16 - *Moon Secrets* (advanced): Scene 1 outline

2. Select *Child* actor, and in the *Properties* tab open *Animations*. To create a new animation for the actor, do the following:
    a. Press the ***(+) button*** just below the list, or select the *Animation* > *New animation* menu. An empty new animation is added to the list:

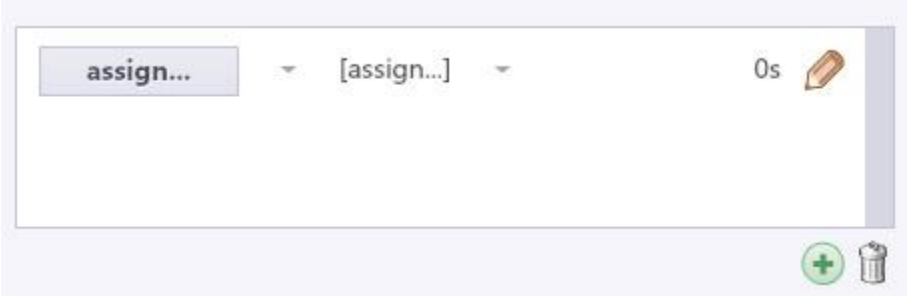

Figure 17 - New animation (unassigned)

    b. Change the triggering event of the animation to **ENTER**. To do this click the drop-down menu on the left and select the ENTER option. This way the animation is activated at the start of the scene. Automatically, the *activation state* of the animation will change to **Default**. It should look like this:

Figure 18 – New animation (assigned)

    For more information on Events and States, consult the *How do animations work in PlayTales® Builder™?* section.
    c. To edit the animation, press the ***pencil button***, double-click on the animation in the list or select the *Animation* > *Edit animation* menu. The **animation edition menu** will appear:

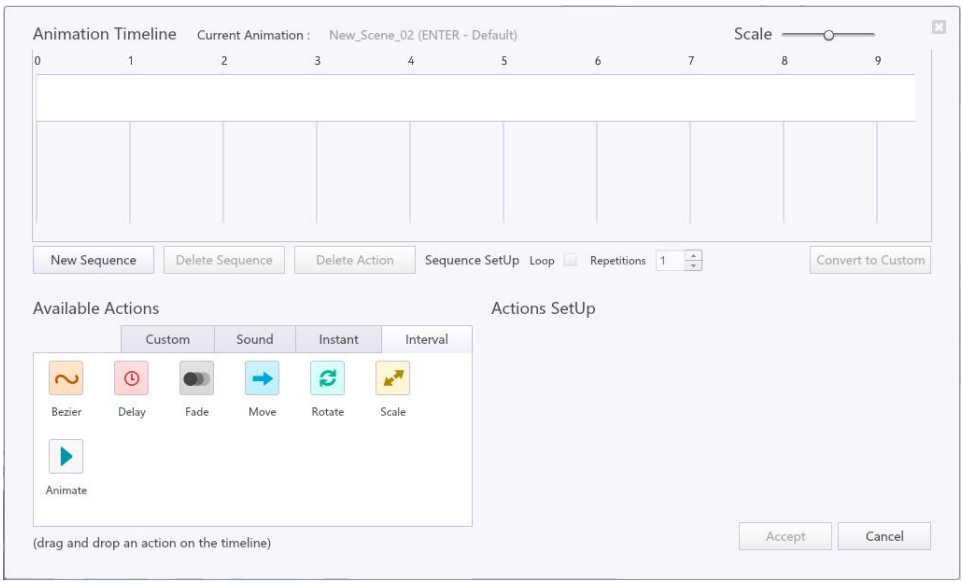

Figure 19 - Animations edition menu

3. In this menu, you can add the **actions** that make up our animation. In our case, we want the actor to move slowly from right to left as if it's floating on water. We will use the **Move** actions to simulate this movement.
    a. Create a new action sequence if one hasn't already been created by selecting the *New Sequence* button. An empty block will appear in the timeline of the animation.
    b. In the **Interval** Actions tab, drag a **Move** action onto the *timeline*. In the *Action SetUp* section a form will appear that we can use to edit its properties.

Figure 20 – Move Action

Enter the following information:

i. Action: **Move**
ii. Type: TO
iii. Duration: 3000ms
iv. Position: X = 700, Y = 125
v. Smoothing function: EaseNone

c. Drag another **Move** action behind the first one on the timeline and enter the following values:

i. Action: **Move**
ii. Type: TO
iii. Duration: 3000ms
iv. Position: X = 335, Y = 125

v. Smoothing function: EaseNone

To set the movement so that it repeats continuously, check the *Loop* box in the properties sequence, just like in the example that is show below:

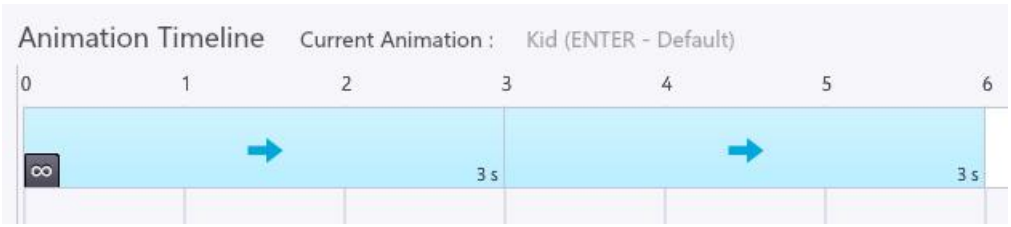

Figure 21 –*Kid* (ENTER – Default) animation

Press *Accept* and the changes we made will be saved in the animation sequence.

NOTE: You can also change the duration of the action by stretching the action in the timeline from the right side wall.

If we open the emulator now, the *Kid* actor should be moving slowly from left to right as soon as the scene starts, and actors *Ripple_2* and *Ripple_1* should also be moving in accordance to *Kid* actor's animation! There is a lot of potential for actor animation with the parent-child hierarchy option.

4. Now, we will also add an entrance animation to the ripples (*Kid* actor's children), to complete this first scene. Select the *Ripple_1* actor and create a new animation by selecting **ENTER - Default**, the same way we did before. Open the Animation editor and do the following:
   a. Create a sequence if there isn't already one.
   b. Drag a **Fade** action onto the timeline.

Figure 22 – Fade Action

And enter the following values:

i. Action: **Fade**
ii. Type: OUT
iii. Duration: 2000ms
iv. Smoothing function: EaseNone

c. Add another **Fade** action to the timeline and enter the following values:
   i. Action: **Fade**
   ii. Type: IN
   iii. Duration: 2000ms

iv. Smoothing function: EaseNone

d. Drag a **Delay** action.

Figure 23 – Delay Action

And then add the following values:

i. Action: **Delay**

ii. Duration: 1000ms

e. Again, set the movement so that it repeats continuously; check the *Loop* box in the properties sequence.

Once you have done this, this animation should look like the following:

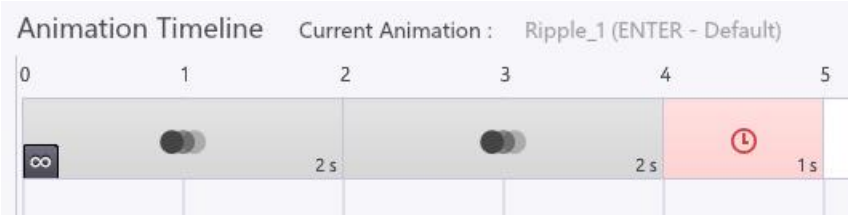

Figure 24 – *Ripple_1* (ENTER – Default) animation

5. Let's create a similar animation for actor *Ripple_2*. In this case, instead of re-creating the animation from scratch, we will copy the animation from *Ripple_1* and paste it on *Ripple_2*, by doing the following:
   a. Select animation **ENTER - Default** we just created for actor *Ripple_1*.
   b. Press **Ctrl + C** (or **Command + C** on Mac) to copy the animation. You can also choose the *Edit* > *Copy Animation* menu.
   c. Select actor *Ripple_2* and press **Ctrl + V** (**Command + V**) to paste the copied animation. A new unassigned animation will appear on the list.
   d. Before editing it, you have to reassign the Event and State. Just like before, we assign the event to **ENTER**, and the state will change to **Default**.
   e. Open the editor and drag the **Delay** action that is listed last in the sequence, and place it in the first position.

The animation should look like this:

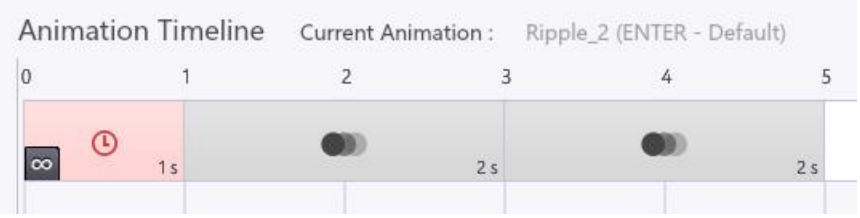

Ilustración 25 – Animación de *Ripple_2* (ENTER – Default)

Open the emulator again to see the result.

As you can see the children actors are running an animation and are affected by the animation that their parent actor has. Thanks to this technique we can easily compile very complex and colorful animations from simple actions and a little imagination.

If you want to see examples of advanced animation sequences, please see the Examples section of the Forum: www.playtalesbuilder.com/forum.

Scene 2:

In this scene we want characters to feature animations when they're touched. We also want to add some sounds as part of their animation, so first we need to add the sound resources to the project.

1. In the Project **Library**, open the **Sounds** tab and press the ***(+) button*** to add new sounds to the project. Search in the *sounds* folder of downloaded resources and select the *02_window_sound.mp3* file to add it. After adding the file set its Type of use as **SFX** (sound effect).
2. Select the actor *Window*, and open the Properties tab to modify the **Anchor** values to X=0 and Y=0.7. We do this because later on we want the animation to have a different effect on the actor.
3. Create a new animation **TOUCH - Default** for the actor, and open it for editing. In this case we will make the actor move in two ways at the same time, running two parallel actions simultaneously. PlayTales® Builder™ makes doing this quick and easy, just follow these instructions:
    a. If no sequence already exists, click the *New sequence* button to create a new one.
    b. Press the button again to create a second sequence. The actions that occur in the same instant are executed in parallel.
    c. Fill the sequences in this order:
    Sequence 1:
        i. Action **Play Sound**:

1. Sound: 02_window_sound.mp3

ii. Action **Scale:**
   1. Type: TO
   2. Duration: 500ms
   3. Scale: X=0.6, Y=1

iii. Action **Delay:**
   1. Duration: 1000ms

iv. Action **Scale:**
   1. Type: TO
   2. Duration: 1000ms
   3. Scale: X=1, Y=1

Sequence 2:

i. Action **Rotate:**
   1. Type: TO
   2. Duration: 250ms
   3. Angle: -2º

ii. Action **Rotate:**
   1. Type: TO
   2. Duration: 250ms
   3. Angle: 5º

iii. Action **Delay:**
   1. Duration: 1000ms

iv. Action **Rotate:**
   1. Type: TO
   2. Duration: 1000ms
   3. Angle: 0º

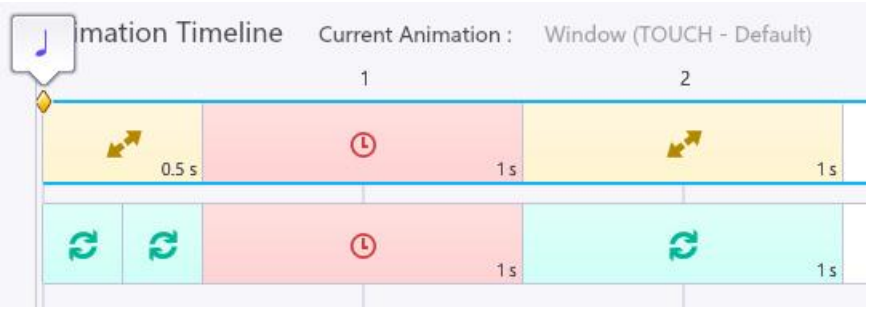

Ilustración 26 – Animación de *Window* (TOUCH – Default)

This way we can create nice animations that active when the actor is touched on the screen. You can launch the emulator to see the result of your work.

Scene 3:

Finally in this scene, we will give the possibility to the actors to be dragged around the screen, and play an animation when released.

1. Add the sounds *03_car_sound.mp3* and *03_toy_sound.mp3*, set them as SFX for type of use.
2. Select the *Car* actor in the *Properties* tab, check the **Draggable** option.
3. Do the same for the *Toy* actor. Also, in the *Properties* tab change the *Anchor* to X=0.5, Y=0.

Once you've set the *Draggable* property, if you launch the emulator you can see now that the actors can be dragged around the screen using your finger. Now to add an additional animation when you finish dragging the characters with your finger, we'll follow the next set of instructions:

4. For actor *Toy* and *Car*, create an animation assigned to the event **DROP**, and the state as **Default**. In the animation editor, add the following settings:
    a. For actor *Car*:
        i. Action **Play Sound:**
            1. Sound: 03_car_sound.mp3
        ii. Action **Move:**
            1. Type: BY
            2. Duration: 500ms
            3. Position: X=-100, Y=-70
            4. Smoothing function: Exponential
            5. Smoothing mode: Out

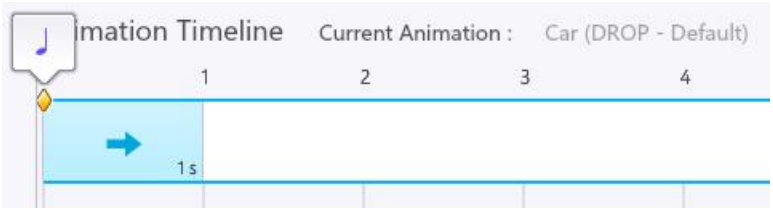

Figure 27 –*Car* (DROP – Default) animation

    b. For actor *Toy*:
        i. Action **Play Sound:**
            1. Sound: 03_toy_sound.mp3
        ii. Action **Rotate:**
            1. Type: TO
            2. Duration: 250ms
            3. Angle: -5º

iii. Action **Rotate:**
    1. Type: TO
    2. Duration: 250ms
    3. Angle: 5º
iv. Action **Spinning:**
    1. Type: TO
    2. Duration: 250ms
    3. Angle: 0º

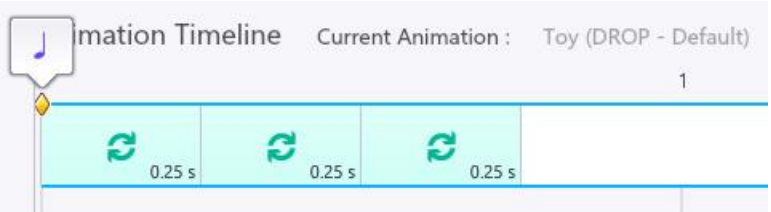

Figure 28 –*Toy* (DROP – Default) animation

So far we've created some basic elements of interactivity for the actors in the scene in a simple and quick manner. To complete our interactive story, let's add the finishing touches.

Background music:

To make our story even more captivating, let's add background music to every scene. To do this, follow these steps:

1. Add *00_background_music.mp3* to the sound library and set its *Type of use* to **Music**.

> **IMPORTANT:** PlayTales ® Builder ™ optimizes sound resources by type, so it is very important to establish that the sound will be used as a Music or Voice sound effect (SFX). For more information on sound resources, please see *Advanced Concepts*.

2. Open project properties by clicking the *Edit* > *Edit Project Properties* menu. The properties tab located in the top right side of the application will show the project properties.
3. Select the Properties tab, and in Background Music, choose the newly added *00_background_music.mp3*.

> **NOTE**: Each scene can have its own background music, and if you want, you can change through actions the music in the middle of a scene. In any cases, you can only choose music files whose *Type of Use* has previously been set as Music.

Now the music we've chosen will play in the background throughout the story. You can launch the emulator to listen to it.

Narrations:

Finally to complete our story, let's add the narrations of the text for every scene.

1. Add the sounds *01_voice.mp3* and *02_voice.mp3* to the sound library and set its *Type of use* as **Voice**.
2. Select the scene and open the *Animations* tab. Create a new animation and assign it to the event **ENTER**, and the state will be automatically assigned to **Default**.
3. Open the animation editor and drag the action **Play sound** from the *Sounds* tab of *Available Actions*.
    a. Assign the *Action Settings* of *01_voice.mp3*.

> **NOTE:** You can assign *Play Sound* actions with narrations to any scene or actor.

4. Do the same for the Scene 3, with the sound *02_voice.mp3*.

With a little effort we are able to add interactivity and motion to our story along with narration, sound effects, and music! We went from a static story on paper to a dynamic tale ready to be enjoyed on a phone or a tablet.

Create a “PlayTales” story

PlayTales® Builder™ offers tons of possibilities for those who want to create really spectacular books. If used properly, the tools available in this application along with your imagination can help you get the most out of your PlayTales® Builder™ experience.

These are just a few examples of what the tools available in Builder can do:

- Create animations by states, so that the characters can have various actions and the book is full of interactivity.
- Create simple puzzles to stimulate user intelligence.
- Make non-linear stories, in the "Choose Your Adventure" style.
- Make mini games.
- And make not only interactive books, but also presentations and applications for companies.

We recommend reading the **¡Error! No se encuentra el origen de la referencia.** section to get the most out of your PlayTales® Builder™ experience.

Test the project on a real device

Although the *Simulator* included in PlayTales® Builder™ is very useful and simulates an almost identical end result, before publishing a project in the PlayTales® catalog or as a standalone application within popular platforms, it is imperative that we check how our story looks and functions on mobile devices and get a preview of what the user's experience will be like.

Luckily, PlayTales® Builder™ also offers tools to **install our project on our physical mobile device**.

NOTE: You will need a Wi-Fi internet connection in order to install your project on your mobile device, and both the computer where PlayTales® Builder™ is installed and the device you want to use to install your story must be connected to the same Wi-Fi network. For more information, please see *Test your project on a real device*.

NOTE: The communication between both ends can be blocked or interrupted by Firewalls and Antivirus programs installed on your computer. Be sure to unlock the network port 8282 to enable Builder communication.

Help: How to open a port in my OS:
Windows: http://windows.microsoft.com/en-US/windows-vista/open-a-port-in-windows-firewall
Mac: http://support.apple.com/kb/HT1507?viewlocale=en_US

1. Download and install the **PlayTales® Builder™ Viewer** application for free on your mobile phone or tablet. With this application we can communicate between PlayTales® Builder™ and your mobile device.
2. From the Builder, open the project you want to test on the device, select the *File > Deploy Project* and choose the device to which you want to install your story. In our case we'll select an **iOS** device (ex. iPhone, iPod or iPad). PlayTales® Builder™ will begin to create the binary information of the project to deploy.

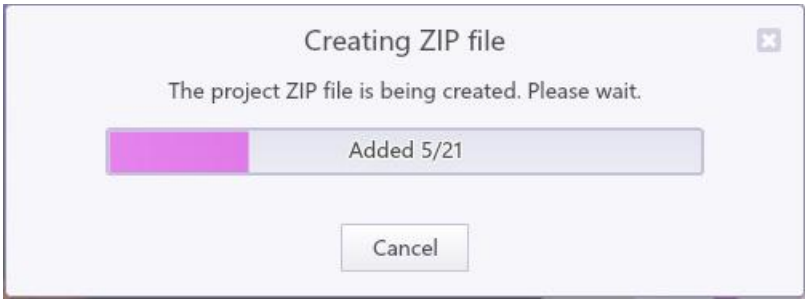

Figure 29 – Deploy a project: Creating the ZIP file

NOTE: You can also deploy a single scene to speed up the process by selecting the *File* menu> *Expand scene* and choose an available device.

3. After compiling the project, a menu will appear in which we must choose the IP address to use for transmitting information. Normally there will be only one option (Local Area Connection or Wireless), but if we are connected to multiple networks or have configured virtual networks there will be several options. Make sure that both the Viewer and Builder transmit and receive information using the same IP address and Wi-Fi network, then press **Connect**.

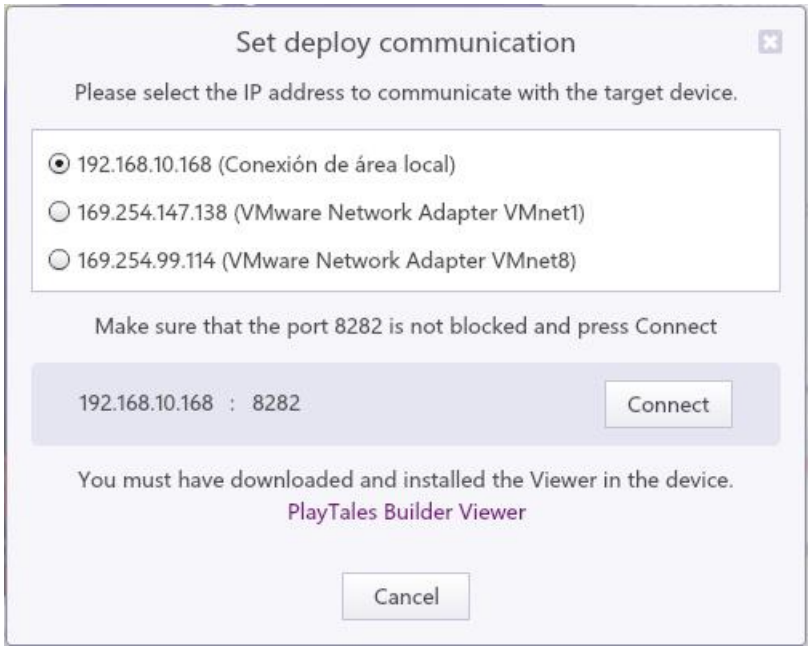
Set deploy communication
Please select the IP address to communicate with the target device.
192.168.10.168 (Conexión de área local)
169.254.147.138 (VMware Network Adapter VMnet1)
169.254.99.114 (VMware Network Adapter VMnet8)
Make sure that the port 8282 is not blocked and press Connect
192.168.10.168 : 8282
Connect
You must have downloaded and installed the Viewer in the device.
PlayTales Builder Viewer
Cancel

Figure 30 – Deploy a project: Select IP

4. PlayTales® Builder™ will then connect to the device to send the information. At this point, press the **Connect** button in PlayTales® Builder™ Viewer.

Figure 31 – Deploy a project: Waiting for device

5. When a device is connected, its name will appear in the Builder and the computer's IP address will appear on the device. Touch the IP address of the device to initiate the connection.

Publish your project

Once you have completed and verified that your project works properly, you are ready to publish your story as a standalone application within various popular mobile platforms.

IMPORTANT: Before you can publish a book, you have to register as a PlayTales® Builder™ user on the official website and purchase a license.

To publish a project from the Builder, do the following:

0. Save your project.
1. Select the *File* > *Publish Project* menu.
2. The *Publishing Wizard* will appear in a new window. To publish your project, you must accept the terms and conditions described in the Legal Publication. Click *Next* once you have read and accepted the legal terms.

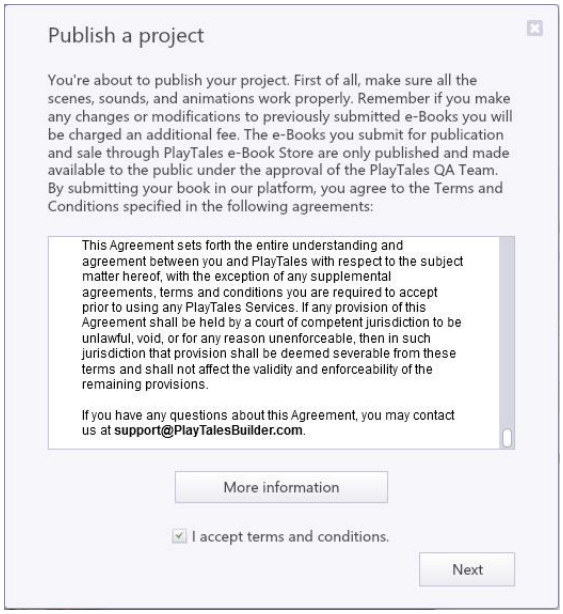
Publish a project
You're about to publish your project. First of all, make sure all the scenes, sounds, and animations work properly. Remember if you make any changes or modifications to previously submitted e-Books you will be charged an additional fee. The e-Books you submit for publication and sale through PlayTales e-Book Store are only published and made available to the public under the approval of the PlayTales QA Team. By submitting your book in our platform, you agree to the Terms and Conditions specified in the following agreements:
This Agreement sets forth the entire understanding and agreement between you and PlayTales with respect to the subject matter hereof, with the exception of any supplemental agreements, terms and conditions you are required to accept prior to using any PlayTales Services. If any provision of this Agreement shall be held by a court of competent jurisdiction to be unlawful, void, or for any reason unenforceable, then in such jurisdiction that provision shall be deemed severable from these terms and shall not affect the validity and enforceability of the remaining provisions.
If you have any questions about this Agreement, you may contact us at support@PlayTalesBuilder.com.
More information
I accept terms and conditions.
Next

Figure 32 – Publish a project: Accept the terms and conditions

3. If you are not identified in the application, the wizard prompts you to identify yourself now. Press the *Login* button and enter the e-mail address and password you chose when you registered on the official website. If you already logged-in, the wizard skips this step.
4. The wizard will now show a screen where you can view and check the properties of your project. Press the *Next* button to begin the publication process.
5. After the delivery, you will see a final confirmation screen that will give you the option to go directly to *My Store* to complete the process.

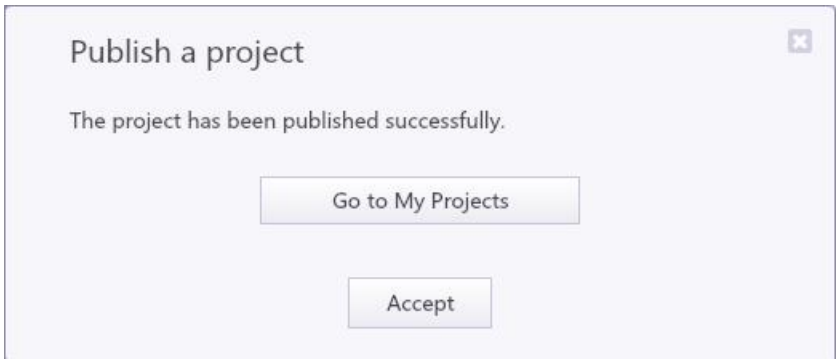

Figure 33 – Project published successfully

Congratulations! You now know everything there is to know about creating and publishing your PlayTales® project! Now just sit back and enjoy bringing your stories to life.

Before using the program

Before we start using the program, it is important to understand some important aspects of the Builder. This will help us understand how and why things work the way they do in PlayTales® and PlayTales® Builder™.

Figure 34 – *Who lives here?*, from Pablo Gastalver

What does it take to make a book in PlayTales®?

If we stop to think about how the downloadable books that are available in the PlayTales® Bookstore are made, we can quickly identify that, if we'd like to make one too, we will have to consider the following aspects:

Ilustración 35 – *Carry, el coche veloz*, from Pablo Gastalver

- **Idea:** PlayTales® books are original and fun, so that a child will enjoy and re-read them many times. It's clear that when making a book you must have a good idea or be able to adapt an existing story.
- **Graphics:** You'll also need to develop and design everything you see on screen: characters, settings, texts, buttons, etc.
- **Sounds:** And we can't forget all the sounds too: narrations, music, and sound effects are also an important part of the story.
- **Programming:** Finally, we need this last element to combine and harmonize all the different creative aspects to create a coherent and interactive story that can be enjoyed on a tablet or phone.

A creative person who wants to create a book could complete the first three creative aspects solo (or with some help), but the fourth part (Programming) seems like a whole lot of work to do for one person.

PlayTales® Builder™ has been created with the ultimate goal of facilitating the task of programming interactive stories and applications, making this book market available to everyone without the need of prior programming knowledge.

PlayTales® Builder™ trusts that the developer will provide the idea, graphics, and sound, and PlayTales® Builder™ in turn will provide simple, intuitive tools to schedule, organize, test, and publish your book.

What is the structure of a book in PlayTales ® Builder ™?

Once we know what is needed to make a book in PlayTales®, we will first begin using PlayTales® Builder™ by analyzing what are the parts that make up a book and how are these components represented in the program.

In a conventional interactive digital story the following elements often appear in each scene:

1. **Narrative:** in scenes we normally find superimposed text that tells what is happening or sample dialogues of the characters, like in a traditional paper book.
2. **Characters and objects:** we see characters and objects that the main character of story interacts with.
3. **Scenes**: the characters live in beautiful and amazing worlds.
4. **UI Elements:** In addition, there are other elements that allow us to navigate between scenes, jump directly to a part, enable or disable the sound, and generally change the normal flow of the story.
5. **Interaction options**: and to take advantage of the opportunities offered by modern smartphones and tablets, we can give the user the ability to interact with characters, setting, texts, and everything else featured in the story.

PlayTales® Builder™ incorporate all these elements, simplifying them to facilitate the task of organizing all the features included in your project. The following graph shows the hierarchy of a book in PlayTales® and the terminology that we use and repeat throughout the documentation to refer to the various parts.

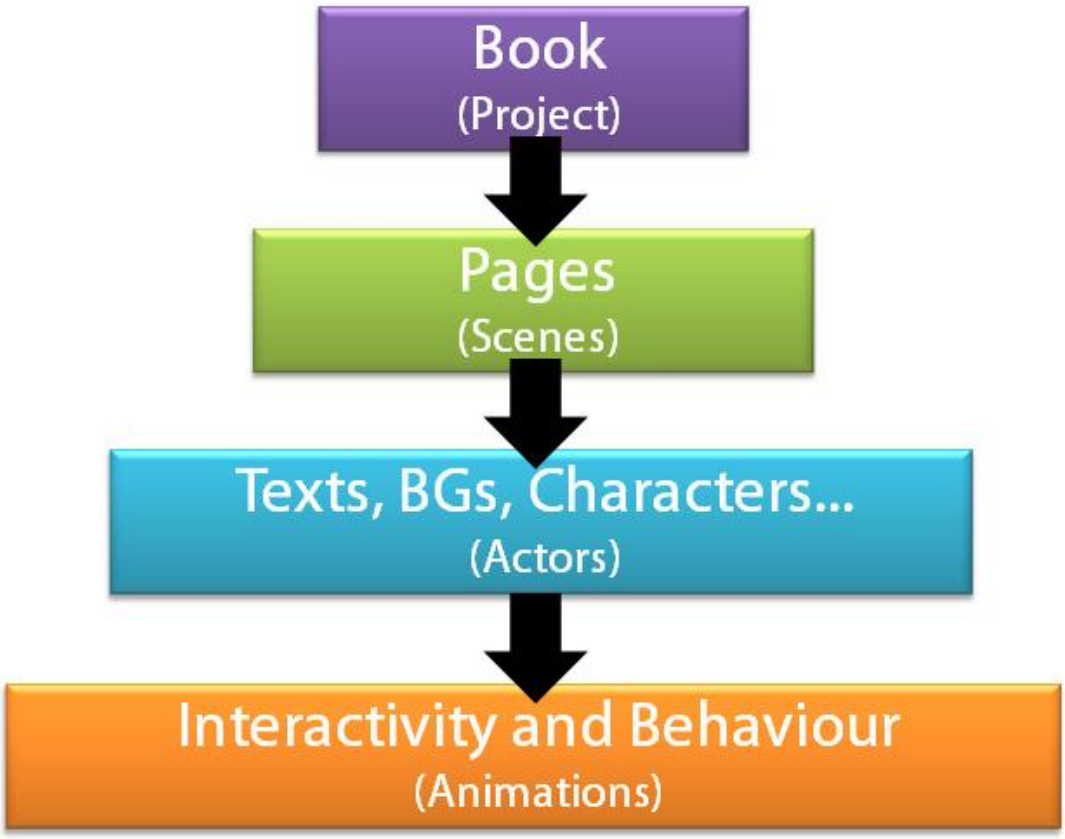

Figure 36 – Hierarchy of a PlayTales® book

- **Book (Project):** a "book" corresponds with a *Project* in PlayTales® Builder™. A project contains all the scenes and all the necessary resources.
- **Pages (Scenes):** a book consists of many pages that make up the entire story. Each "page" of the book corresponds to a *Scene* in the Builder.
- **Text, Backgrounds, Characters, Buttons, etc. (Actors):** all graphic elements of the pages; narrative text, background objects, characters, etc. are all called *Actors* in the Builder, without any distinction.
- **Interactivity and Behavior (Animations):** the movement of the actors and their reactions to user interaction are shown in PlayTales® Builder™ as *Animations*.

Current Limitations of PlayTales ® Builder ™

Finally, before you start creating a book, you should know and understand some characteristics and limitations that currently exist within the PlayTales® engine. Keep in mind however, the engine is continually evolving and it is likely that any limitations will soon disappear.

To keep up with the latest updates, visit the official PlayTales® Builder™ website: www.playtalesbuilder.com.

- **The page layout is always horizontal.** All PlayTales® books have a landscape or horizontal format.
- ***Scenes* are independent.** Actors do not persist between scenes. You can't use something that is defined in one scene and then use it again in another. Everything lives and dies in the same scene.
- **Everything is an Actor.** Although mentioned before, it is important to keep in mind that although it is common of you to think a project of this type dividing the elements into setting, characters, texts, etc., PlayTales® Builder™ identifies everything in a scene as an *Actor*.
- **No randomness.** Actors' behavior when responding to events at a given time will be the same. But although they can't implement a random animation, they can respond in one way or another depending on their state. More information about states in *Animation States*.
- **The behavior of the actors is based on Events.** As indicated, all animations are triggered by a certain *Event*, such as touching the Actor (*TOUCH*) or dragging it around the screen (*DRAGGING*). There are also *Events* that are triggered without user interaction, such as when the scene starts (*ENTER*) or during *Autoplay* mode (*AUTOPLAY*). The animations will run when they are triggered by the *Event* you select, and the Actor is in the appropriate *State*. Click here for more information about **¡Error! No se encuentra el origen de la referencia.**.
- **The graphs are based on properties Cocos2D**. If you have worked with any paint, editing, or digital animation program you will find many similarities between these programs and PlayTales® Builder™ in regards to how they work. However, there are a couple of differences that are important to consider:
    - **The point of origin** in the canvas of the scene is in the **lower left corner**. The X coordinate increases to the right (from 0 to **1024**) and the Y coordinate upwards (from 0 to 768).
    - **Child actors** of another *Actor* **are positioned from the bottom left corner of the texture of its parent**, and not from its *Anchor Point* as expected.

Basic Concepts

In this section we explain all the concepts you need to know to create a static book, like the ones we are accustomed to seeing on paper.

If this is your first time using PlayTales® Builder™ or if you're still confused about some advanced concepts, take a look at this section before getting started.

Figure 37 – *Willy and The White Room*, from Bryan Krupkin

Create a project

Okay, the first thing we'll do is *Create a new project*. When you open the application, the first thing you should see is the welcome screen.

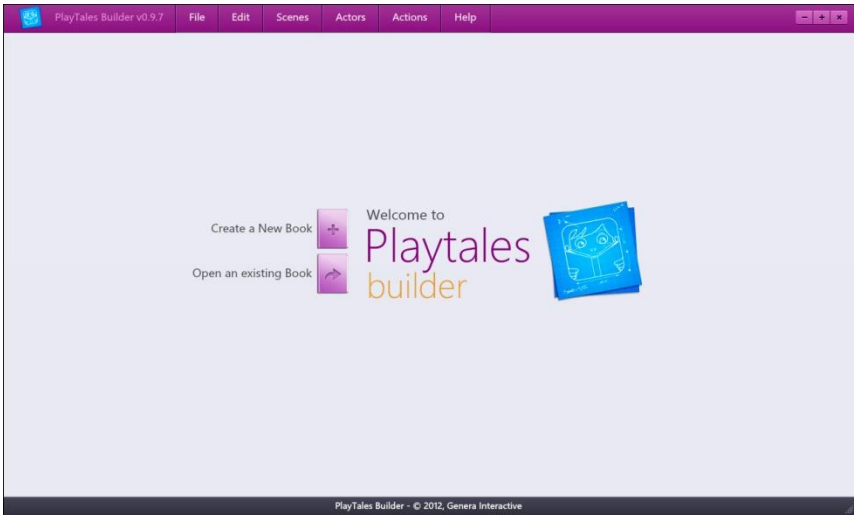

Figure 38 – Welcome screen

In this screen you can click the *Create a new project* button. We can also select the *File* > *New Project* menu. Once you do this, the following window with the fields described below will appear:

Figure 39 – New Project window

- **Location of the project:** In this field you must enter the absolute path of the directory on disk where you want to create the project. (For example: *C:/Builder* on Windows, or */Users/UserName/Builder* in MacOS). We can press the **Search** button to find your way easily through the OS file browser.
- **Project Name:** Here we'll introduce our project code name. The directory name and project file names will match this value.

IMPORTANT: We do not recommend using spaces or unusual characters in this field. Try to only use letters and numbers.

IMPORTANT: Neither the project name, nor the publication name will be the final title of the digital book you'll publish; they're just names to identify it. You can set the final name within the web settings later on.

Once you press OK, a folder will be created with the name specified in the *Project Name*, and within this same folder the configuration file (.*tbbuilder*) will also be created and other folders that contain the files and other useful resources. The newly created project will open and within the work area the first scene should be visible. For more information on the work area, consult the *workspace*.

Add resources to your project

To start working on a project in Builder, the first thing to do is add the graphics and audio resources to the *Project Library* so we can use them throughout the project. The library can be accessed at any time by selecting the *Library* tab located in the lower right section of the application. Within this tab, you can navigate between the different resources grouped by type.

PlayTales® Builder™ differentiates resources used in the project as so:

- **Images:** Here you'll find the image files that we can use as a *Texture* of the actors or to make up the *Spritesheets*.

> **NOTE:** A *Spritesheet* is a sequence of images all of them in the same file, that represents one or more animations.

- **Sounds**: In this part we can see the audio files used as *Music*, *Narration*, or *Sound Effect (SFX)* in the project.

> **IMPORTANT:** It is crucial to choose the right type of file usage depending on how you will use the sound. You should carefully read the *Advanced Concepts* section to understand how to choose the *Type of use* correctly and optimize the sounds of the project.

- **Spritesheets**: to optimize *Frame by frame Animation* of the actors in the project, PlayTales® Builder™ works with Spritesheets that reduce memory load of the graphics that form each frame. In this section the *Spritesheets* are available in the project for use on the ***Animation* actions**.

  All the *Spritesheets'* resources (and also the *Particles*) consist of two files with the same name, which contains the graphic (*PNG*) and the configuring resource (*PLIST*). **The two files are needed for the resource to be used correctly.**

> **NOTE:** To **create a new *Spritesheet***, view the group of Images, select the files you want to be the *frames* of the *Spritesheet,* and press the pencil-shaped button.

- **Particles:** to add more attention grabbing features to the projects created with PlayTales® Builder™, we have incorporated a particle system that allows you to easily create animation effects such as fire, a fountain, a column of smoke, rain, snow, an explosion, etc. Effects that otherwise would be very difficult to animate from scratch. In this section we can see the particle resources available in the project.

> **NOTE**: To create particle resources you should use the PlayTales® Particle Creator™ program available for free via the official website: www.playtalesbuilder.com.

In every section, there are controls to **add, delete,** and **filter** the available resources of each group.

The resources that are added to the project are copied to the corresponding folders in the directory / *assets* of the project, so that the changes made to the original files will not be reflected in the copy. You can directly access the added files in different folders / *assets*. In the same way, you can directly add files to these folders through *Windows Explorer* (in Windows) and *Finder* (MacOS) and they will immediately be available in the corresponding group in the *Library*.

NOTE: Remember that in able to use a *Spritesheet* or a *Particle* resource, you need to add both the *PNG* and *PLIST* files that compose them.

Edit Project Properties

The general properties of all the elements of the book can be found within the **Project Properties**. You can view and edit these elements in the editing forms of property, located in the upper right side of the program by clicking on *the Edit > Project Properties*.

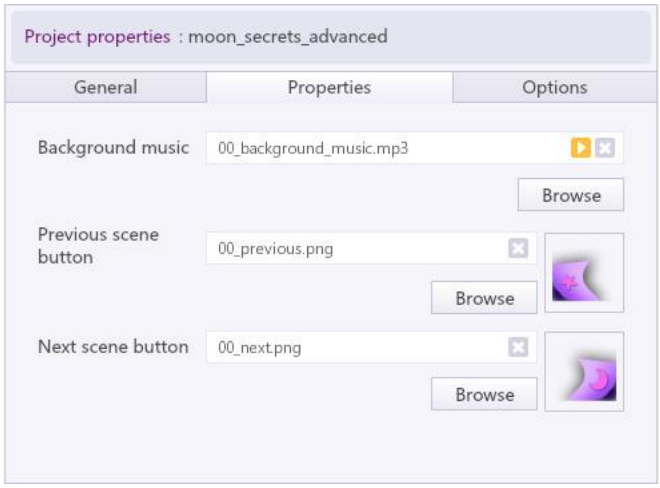

Figure 40 – *Project properties* example

The project properties are grouped into several categories and defined as follows:

General:

- **Location of the project:** we have indicated when we created the project. It cannot be changed afterwards.
- **Project Name**: we also indicated the name earlier and it cannot be changed.

Properties:

- **Background music:** here we can choose a music that will continuously plays throughout the project's *Scenes*, without cutting steps in between one scene to the next. To add music to your project you must press the *Search* button and select an audio file whose use has been set to music.
- **Next and Previous scene buttons:** you can choose what graphics you'd like to use that appear in the lower left and right corners and serve as portals from one page to the next, just like the ones in this example.
  By pressing the *Search* button we can choose any graphic that is available within the graphic resources of the project.

Options:

Cover options: The PlayTales® *Default Cover* is actually a scene from the book that is automatically entered as the first scene and serves as the book's main menu. In this scene you can choose a graphic like the *Cover* of the book and choose your preferred *Reading Mode* (Read to me, Read by myself, Autoplay, etc.).

Figure 41 – PlayTales® *Default Cover* example

By checking the option **Use Default Cover** you'll enable the controls to use the PlayTales® *Default Cover.* If you want to use your own cover, and not the cover of PlayTales®, this option should be left unchecked. Once selected, the following options are enabled:

- **Include in the scene selection menu**: If this option is enabled, the default cover page will be included in the scene selection menu to be accessed directly from the PlayTales® menu. If you do not want to give direct access to this scene, you have to disable this option.
- **Include "Autoplay":** if your project includes the *Autoplay* feature, which reads and animates the story automatically, you can select and activate this option from the *Default Cover*. Otherwise, the option will not be displayed in the book menu. For more information on *Autoplay*, see the *Advanced Concepts* section.
- **Include "Read by myself"**: as with the *Autoplay* option, if your project implements reading modes *Read to me* and *Read it myself* and you want people to be able to select the Read it myself mode from the *Default Cover*, this option must be enabled.

NOTE: the ***Read to me*** option is always enabled in the Default Cover

For more information on reading modes, see the *Advanced Concepts*.

- **Cover**: In this field you can specify the image you want to be displayed as the cover of the book. The image must have a size of 397x525px.
- **Cover Preview**: Pressing this button shows a preview of how the cover will appear with the options selected above.

Create a scene and edit its properties

Each page of our book will correspond to a scene in our PlayTales® Builder™ draft. To create a new scene click the *New Scene* button on the toolbar or through the *Scenes* > *New Scene* menu.

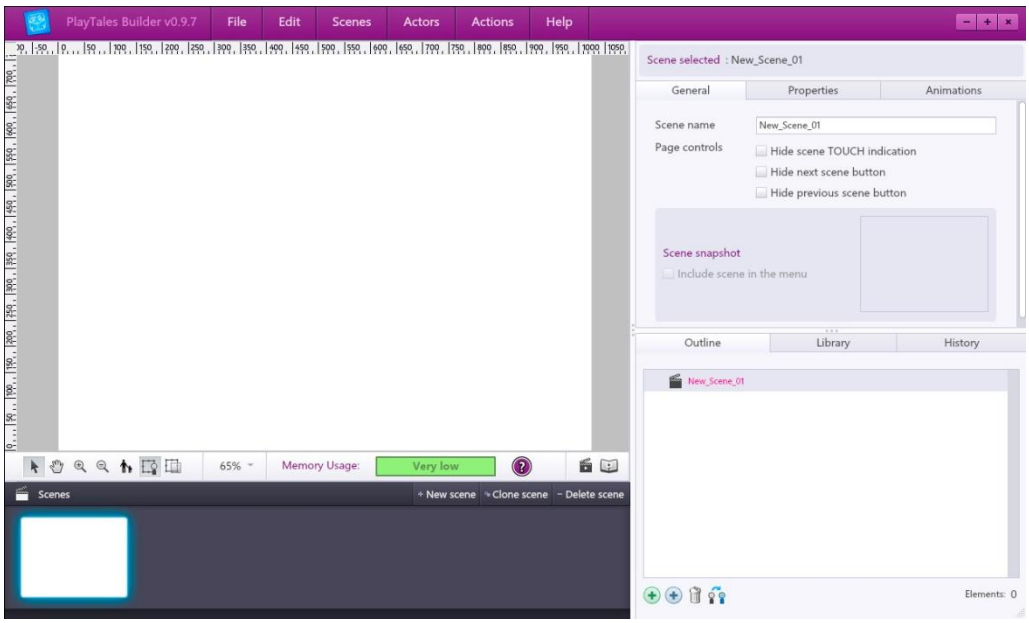

Figure 42 – New scene

The newly created scene will be added to the list of project scenes and it will appear in *Scenes Carousel* located in the bottom of the program. You can automatically select these scenes and edit their properties in the forms located in the upper right side of the program.

For this scene, we can edit the following properties:

General:

- **Name of the scene:** here we can choose the name of the scene. Choosing a good name for the scene will help us to identify it later during importations, or when receiving support from the PlayTales® team.
- **Controls page:** in this group can enable or disable the following aspects of the page:
    - **Hide Touch indication:** if you enable this option the *PlayTales® Default Touch Indication*, which indicates to users which actors are touchable, will not be shown in the scene. Enable this option if you're going to use your own indication or you don't want to show any.
    - **Hide previous scene and next scene buttons:** With this option, you can hide the go to previous and next scene graphics, which are defined in the project properties (see **¡Error! No se encuentra el origen de la referencia.**) for one particular scene. Use this option if you don't want the user to return to the previous scene or advance to the next,

or if the sequence of scenes in your book is not linear, or if you don't want them to be able to go from the first scene to the last scene of your project or vice versa.

- **Include scene in the menu (for quick access to scenes):** If you enable this option, a screenshot taken from the scene will be included in the *Scene selection menu* and can be accessed directly from there. The screenshots are taken from the PlayTales® Builder™ *Simulato*r by pressing the **camera button**.
  You can find more information on the PlayTales® Builder™ *Simulator* here: **¡Error! No se encuentra el origen de la referencia.** *Simulator.*
  Screenshots of the scenes are stored in the */assets/snapshots* of the project with the same name of the scene in the project. If you change the name then you have to retake the screenshot or rename the file in the directory.

Properties:

- **Background color:** in this field you can edit the background color of the scene by selecting a color from the drop-down menu or by typing its hexadecimal code directly.
- **Background music:** similar to the project itself, each scene can play specific background music, replacing any background music that was being played. This characteristic is equal to its namesake in the project.

> **NOTE:** If you replace the background music in a new scene, the above is not recovered on leaving the same, if not already is from the new change that you post. This also happens with the change action background music.

- **Fade entrance of music (Fade-in):** in this field you can indicate whether you want the new music of the scene that replaces the previous music starts playing within a specified time.

Animations:

In this section you can specify which animations you want to apply to the elements within the scene. Later in the block *Animations* we will deeply analyze the animations of scenes and actors.

Create Actor and edit its properties

All elements of the book's pages will correspond with *Actors*, no matter if they are characters, background elements, texts or interface elements. Actors can be created in two ways:

1. By pressing the *Create New Actor* button in the **Outline** section located in the lower right corner of the program, or via the *Actors > New Actor* menu.

> **NOTE**: If you press this button when an actor in the scene is selected, the new actor that is created will be the *Child* of the selected actor. You can find more information on *Parent* and *Child Actors* in the *Advanced Concepts*.

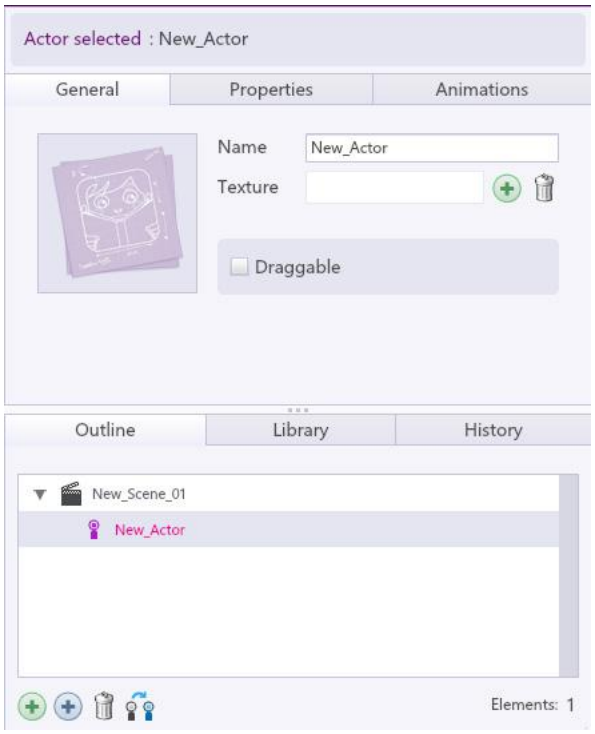

Figure 43 – New actor

2. Dragging an image directly from the *Images* section of the project **Library** to the Scene Canvas. The newly created actor will have the texture of the image that was dragged.

Once you've created and selected the actor, as usual, its properties are displayed in the forms on the top right of the program. This is where you can set the following properties:

General:

- **Name:** just like what we did with the *Scene*, it is important to identify the actor with a name because later on it will make it easier to reference it in *Actions* and other parts of the project.
- **Texture:** corresponds with the image that shows the actor in the scene. You can assign it an image from the *Images Library* by pressing the **(+) button**.
- **Draggable:** This property indicates that the user can **drag** the actor on the screen of a mobile device using their finger. If enabled, the user can move the actor on the screen while reading the book, and also enable the actor to receive *Drag & Drop Events*.

Properties:

In this section you can configure all the graphic properties of the actor in the scene, such as position, size, rotation, etc.

- **Anchor point**: used to determine the reference point of the texture that will be the reference for the rest of the graphic properties of the actor. Specifically, the *Anchor Point* is used for the following:
    - Positioning
    - Rotation
    - Movement

  The values specified in X and Y are related to the size of the texture of the actor. So, if for example we set X=0.5 and Y=0.5, the anchor point will be placed at the center of the texture, X=0 and Y=0, locate it in the lower left corner, X=1 and Y=1 situate it in the upper right corner. Values below 0 and above 1 can be used to situate the point beyond the limits of the texture.

IMPORTANT: Remember that the positioning of the C*hild Actors* is not made from the anchor point of the *Parent Actor*, but from the lower left corner of their texture.

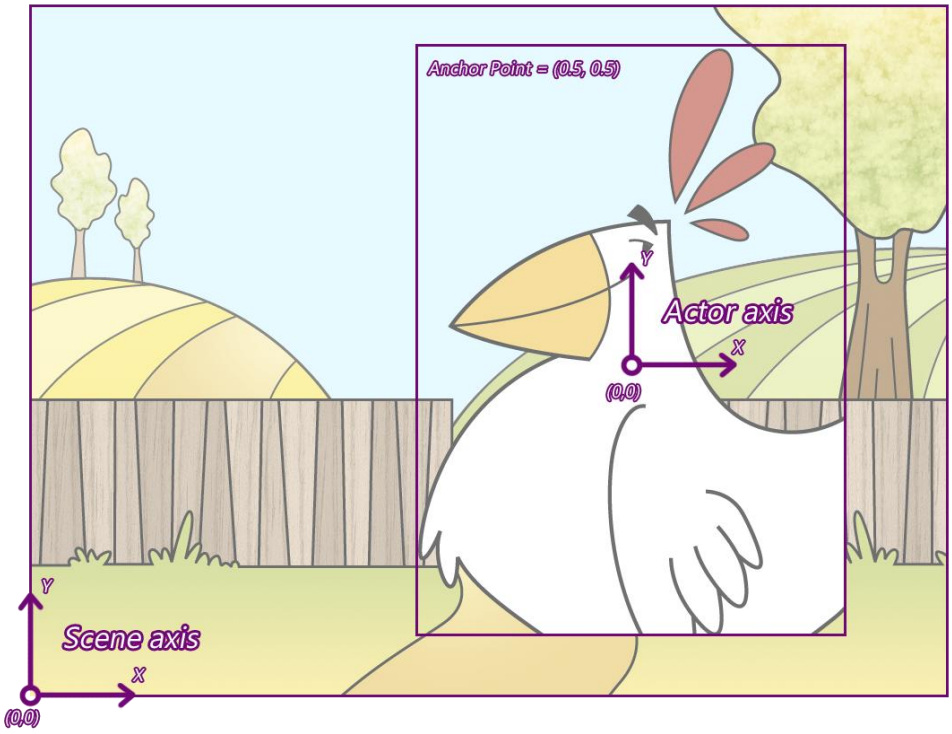

Figure 44 – *Anchor Point* example

- **Position:** indicates the position of the actor in the scene (or within the coordinate system of the parent, if it is its child).
- **Scale:** sets the relative size of the texture of the actor. The original size of the texture corresponds with the values X=1 and Y=1 of the *Scale*. Use values below 1 to make the texture smaller or above 1 to enlarge it.

> **NOTE:** If you are using negative values the texture will be reversed in the corresponding coordinate axis.

- **Rotation**: rotates the texture of the actor. The values go from 0º to 360º.
- **Alpha**: indicates the transparency of an actor's texture. The values range from 0 to 1, with 0 being fully transparent and 1 fully opaque.
- **Z-Index**: with this property, we can control the positioning of the actors in the Z axis. This is used to determine which actors will be above others when their textures overlap in the scene. The higher the value signifies that the actor will be more "on top of" other actors. *Z-index* is a very powerful property but it is important to understand it very well.

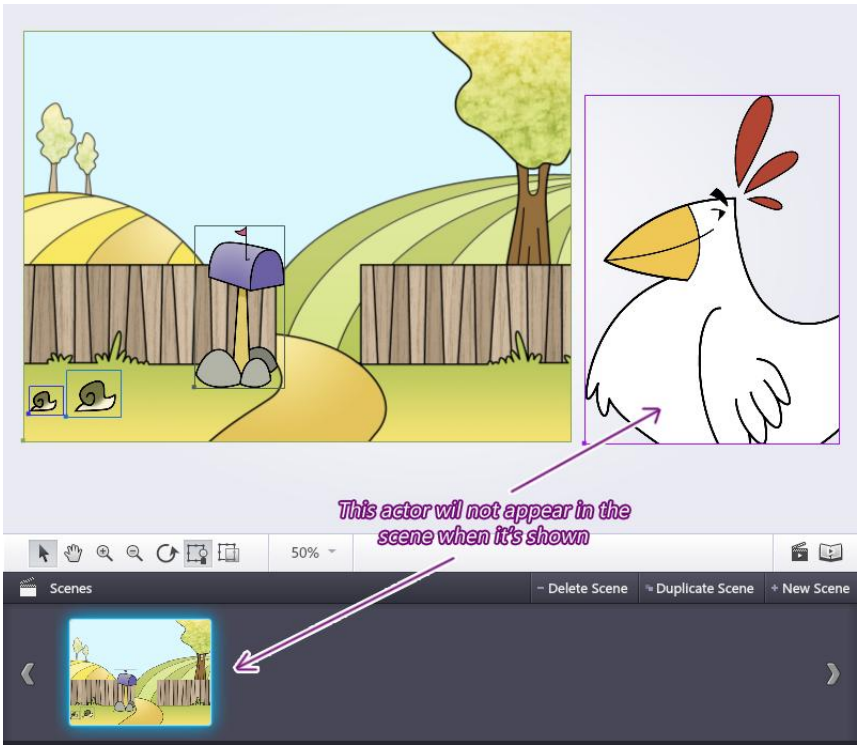

Figure 45 – Scene example (*Cockadoo the Rooster*, from Raquel Mayoral)

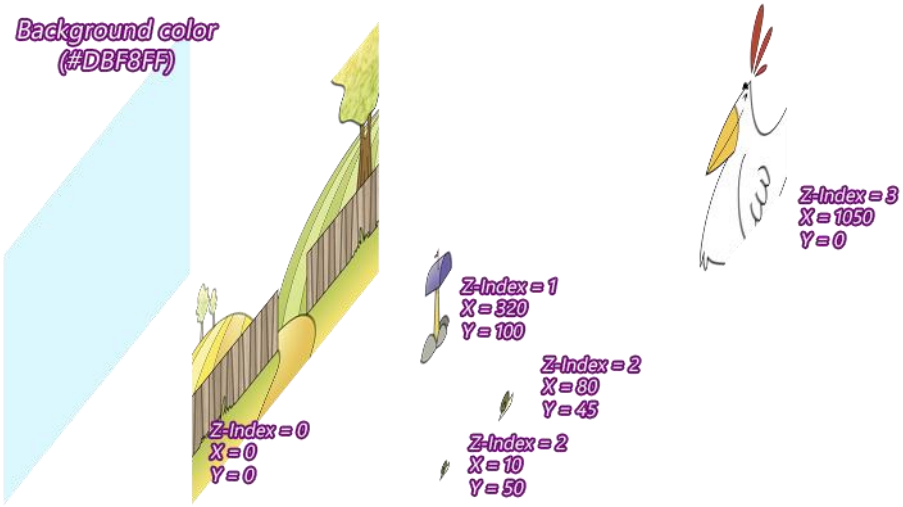

Figure 46 – Z-Index sorting example

- **Bounding Box:** and finally with this parameter we can specify what we want the area of *Actor* interaction to be, both with the user and with other actors. In some cases, especially when the

actor has a very large texture or many transparent parts, we don't want the user to only be able to touch a small part. With this property we can define with a rectangle that part.

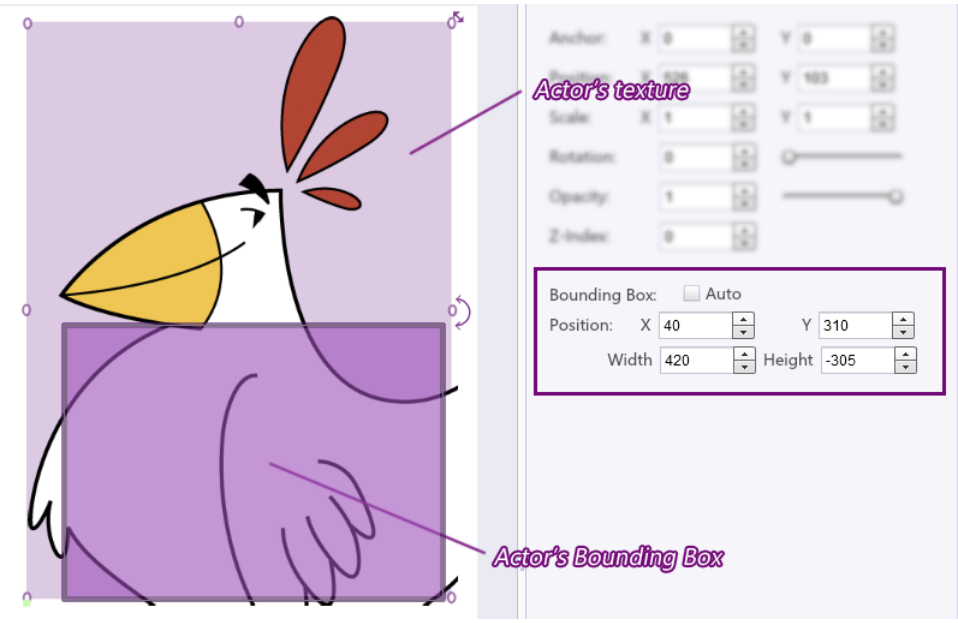

Figure 47 –*Bounding Box* example

Thanks to *Bounding Box* then we can control the areas of:

- User Interaction of *Touch* and *Drag* the actor.
- Actor Interaction with other actors in the *Drop End* Action.

With X and Y we can specify the relative position from the *Anchor Point* of the actor, and with the *Width* and *Height* the size of the box. If we select **Auto**, the *Bounding Box* will automatically adjust to the size of the actor's texture.

NOTE: The *Bounding Box* can also be modified visually in the *Scene Canvas* by pressing the ***Show/Hide boundings* button** in the taskbar.

Animations:

In this tab, just like the *Scene* properties, you can create *Animations* that give the *Actor* behavior. All you need to know about *Animations*, *Events* and *States* can be found within the *How do animations work in PlayTales® Builder™?* section.

Launching PlayTales® Simulator

With all these different animations and designs, it is crucial to see how the effects and changes you've made to your project appear as you develop it. Other programs that are similar to PlayTales® Builder™ only incorporate tools to "send" your project to the device to solve this problem, but also this is necessary to preview the final result on the device where your books will be used, normally the process of doing this is slow, difficult, and you need to have a physical device in order to view the project.

PlayTales® Builder™ not only includes tools to deploy your projects to physical devices for testing before publishing, but also makes it much easier for the testing of the project by **including a Simulator** that lets you preview the book instantly, just by pressing a button without additional hardware or software.

To launch the *Simulator*, press the buttons located on the bottom-right side of the toolbar to launch it *Tablet* size or *Phone* size. You can also select the *Edit* > *Emulate Tablet* or *Emulate Phone* menus.

You can easily see if the scene graphics are very large or very small, or if it's easy or difficult to interact with them on these types of devices.

Figure 48 – Simulating *Eco's favourite things*, from Pilar Leandro

NOTE: To launch the *Simulator*, it is necessary to select a scene from the *Scene Carousel*. The *Simulator* will launch from the selected scene.

The *Simulator* incorporates controls to *close*, *restart* from the selected scene, or take a *screenshot* to include that scene in the Quick Access Menu scenes of PlayTales®.

When you take a screenshot of the scene in the simulator, in the *Scene Properties* the Scene Snapshot control menu will become enable, which you can find in the *General* tab. If you enable the **Include Scene Menu**, you will be able to select the current scenes from the *Select Scene* menu in the PlayTales® control panel.

Figure 49 –PlayTales® *Select Scene* menu

And that's all you need to know about the basics of PlayTales® Builder™. As you start to use the program you can see how easy it is to create a static book, like the paperbacks we are accustomed to, using the Builder. Now, let's seize the opportunities offered by new technologies and convert our book into an interactive story. In the next block we will see how we can create animations and give life to the elements in each scene.

Animations

In this text we will discuss in depth how to create and edit animations for the *Actors* and *Scenes* of your story. Animations allow us to give **movement** to our characters and scenes, and also create **interaction** with the user. Furthermore with animations we can control various different features, like the transitional flow of scenes or the reproduction of music and sounds.

Animations in PlayTales® Builder™ can be quite complex and difficult at first because of the great potential they have, but with a little practice you'll see all the possibilities they offer. In the **official PlayTales® forum** there are many examples and discussions you can use to learn about all the different tools available in Builder.

http://playtalesbuilder.com/forum/

Ilustración 50 – *Sam does a lot*, from Juan José Jiménez

How do animations work in PlayTales® Builder™?

The animations and interactivity in PlayTales® Builder™ are based on the paradigm of event-driven programming. This means that the animations that we assign our *Actors* and *Scenes* are executed when they receive the **Event** that triggers the Animation. For example, if you create an animation for an actor in the scene and associate the animation with a **Touch** event **(TOUCH)**, this animation is activated when the user **touches** that actor.

NOTE: *Animations* are linked with *Events* and implement **interactivity** into the elements of the scene and give them certain **behaviors.** These terms will be used often in this section because they all have to do with **Animation**.

IMPORTANT: While an animation is running, although the element receive again the event that actives the animation, **it will not run again until the actual execution has finished.**

Animation Events

Actors and scenes can be animated through several *Events*, and gradually we can add more Events that will create more possibilities as you develop your book your book.

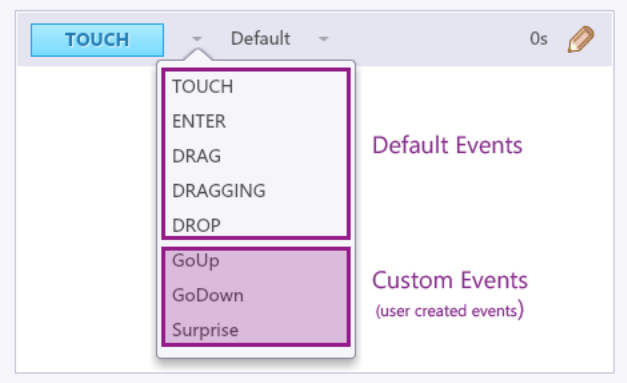

Figure 51 – Events in an animation

Normally, events often occur because the user has interacted in some way with the device, for example they've touched the screen (*TOUCH*), or because they have dragged their finger across it (*DRAGGING*). But there are also events that occur automatically at certain moments, without interaction from the user, such as when a scene begins (*ENTER*).

Specifically, these are the events that can be associated with the animations of scenes and actors:

Events that activate *Scene* animation:

- **ENTER**: When the scene begins, it activates the *ENTER* event. We can use this event when we want the background music to begin as soon as the scene starts, or to activate the narration of the text, or to change the *State* of some actors, etc.
- **AUTOPLAY**: If the book *Reading Mode* is set to *Autoplay*, as soon as the scene begins the *AUTOPLAY* event will be dispatched (instead of *ENTER*). The animations that are associated with the *AUTOPLAY* event will be played only in this mode. You can find more information on *Reading modes* and *Autoplay* in *Advanced Concepts*.

Events that activate *Actor animation*:

- **ENTER**: similar to the events that activate scene animation, each actor's that are assigned an *ENTER* event will animate automatically as soon as the page is opened (that beginning of the scene is the event trigger). In this way, actors will animate automatically and the user doesn't need to do anything to trigger the event.
- **TOUCH**: This is a very common event and the one you´ll probably use the most in your books. The Actors' animation event is triggered when the user interacts with them by touching them with their fingers.
- **DRAG, DRAGGING,** and **DROP**: These three events that can trigger the animation of an actor, basically *Drag & Drop* occurs when the user **drags** an actor around the screen with their finger.

> **NOTE:** In order for an actor's events to be triggered, you must first make sure that the actor has been configured to be ***Draggable***. You can do this in the *General* properties form of the actor, by enabling the *Draggable* option.

Drag & Drop events for an Actor:

If you want to assign a certain behavior to an *Actor* when the user does a *Drag & Drop* operation with it, it's important that you understand how DRAG, DRAGGING, and DROP events are produced.

To better explain how these events are triggered, we will analyze the flow of events that produce the two possible cases in which a user can interact with an actor:

- When the user only touches the actor (not a *Drag & Drop* operation): in the case where the user touches the actor but does not drag it around the screen, the actor is assigned the following events:

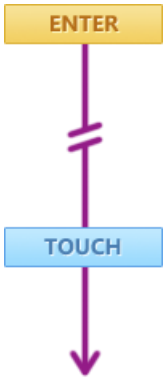

Figure 52 – Events flow: *TOUCH* only

As mentioned before, the actor's *ENTER* event will be triggered at the start of the scene. When the user touches the actors with their finger the *DRAG* event is triggered, this indicated that the *Drag & Drop* operation can begin. However in this situation our user removes their finger without dragging the actor, in this case the actor's *TOUCH* event will be triggered.

- When the user touches and drags the actor: when the user does a *Drag & Drop* operation with the actor, this action will produce the following flow of events:

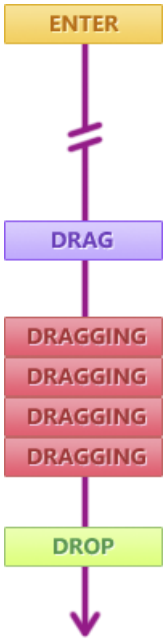

Figure 55 – Diagram of flow of events: TOUCH only (actor dragable)

The first thing, like always, is to receive the *ENTER* event. This event is activated as soon as the scene begins. Next, the actor´s DRAG event is triggered, starting the Drag & Drop operation. Once the user begins to drag the actor with the finger, the actor successively activates

DRAGGING events as the user moves the actor around the screen. In the moment that the user removes their finger from the screen, the operation is complete and the DROP event is activated.

For more information on how events work and to test them directly on your device, download the sample made by PlayTales ® Builder ™ in the *advanced examples* section of the forum: http://www.playtalesbuilder.com/forum/topic/22/.

Animation States

The *Animations*, or the *behavior* of the actors, is determined by the events they receive. So in this scenario the actors always behave the same way when, for example when we *touch* or *drag* them, but sometimes we want the actor to do different things although it receives the same event. To accomplish this, we can associate the animations with a ***State***.

By doing this, for example, we can make a door open or close when the user touches it depending on whether its state was "closed" or "open".

NOTE: If your book does not incorporate animations by states, we can leave the state of all animations as ***Default***.

Animation Structure

The *Animations* of PlayTales® Builder™ are a result of the *behavior* that is "programmed" into the scene elements. One advantage of PlayTales® Builder™ is the complexity of the structure of the animations, which provide you with a wide range of tools and possibilities to give life to the elements of the scene as if it was programmed directly onto the device. The best part about using the Builder you can do this without having to learn the different programming languages for each platform.

Let's take a look at the following example to understand the different parts of an animation:

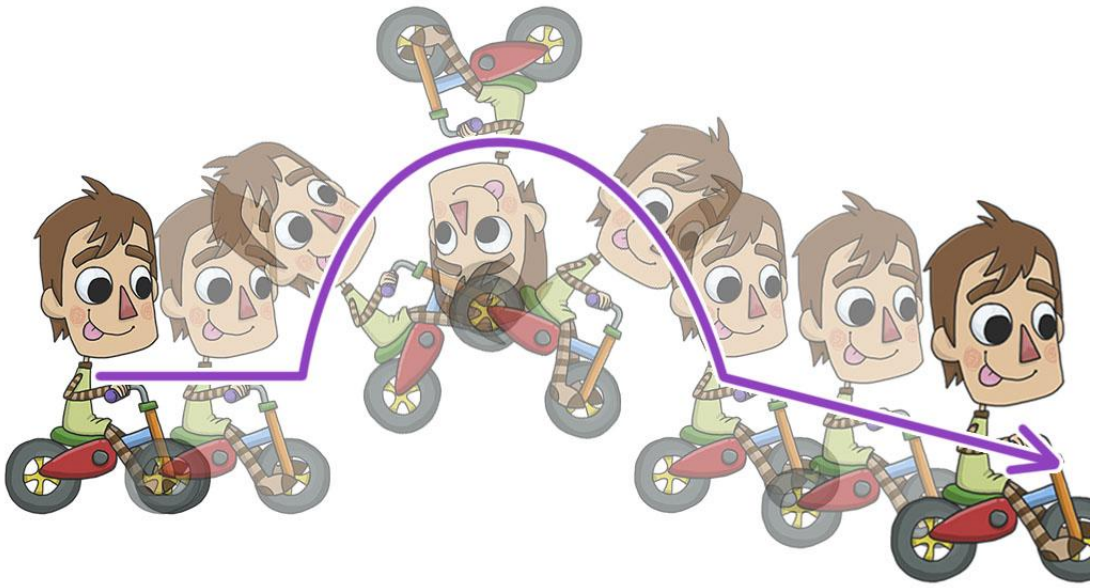

Figure 53 – Complex animation example

In this example, we want our actor to perform a rather complex animation, in which it first moves forward, then jumps while doing a somersault in the air, and falling forward again in a different direction. Furthermore, we want our user to hear sound effects.

An animation can be understood as a ***Sequence of Actions***, that is to say actions such as *Move, Rotate, Play sound*, etc. which occur one after another. In addition, for various *actions* to be running simultaneously at any given time, **the Sequence of Actions can be grouped parallel** in the animation.

In PlayTales ® Builder ™ an animation consists of a set of *Sequences of Actions*, which all run in *parallel*. The minimum unit in the animation is *Action*.

A possible *Animation* PlayTales ® Builder ™ to resolve this would be:

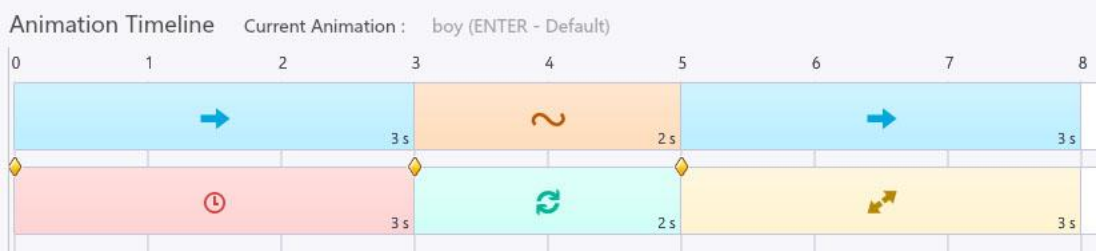

Figure 54 – Possible animation in the Builder

The animation is formed in this case by two sequences. Consider the different parts of the animation:

NOTE: The **yellow rhombus icon** of the animation indicates that at this point we can find a series of *Instantaneous Actions*. To see which ones are a part of this set of actions, push the yellow rhombus icon to open the balloon that contains them.

Before the jump:

In the first sequence the *Move* action is included, which moves the actor forward until the jump. In the second sequence a first action *Reproduce sound* is included, reproducing the sound of the engine, and then continuing with the *Delay* action, which just makes time pass and does nothing.

During the jump:

Add a *Bezier* action to the first section; this causes the actor to make a parabolic curve which simulates the jump. In the second sequence add a *Rotate* action which causes the actor to rotate on its axis backwards while making a curve…completing the movement. Before the action sequence is completed, add a new *Reproduce Sound* that makes the sound of an engine again.

After the jump:

The first sequence will include a *Move* action that moves the actor outside of the screen view, moving slightly in a downward direction. Meanwhile, the second sequence includes a *Scale* action, the size of the actor will change a little, giving us sensation that it is moving closer.

You can find the resources and the complete example at: http://playtalesbuilder.com/resources

Actions

After you understand the Animation Structure, we will list all the *Actions* available in PlayTales Builder and explain its most relevant properties.

To better organize the actions in the *Edit Menu* of an Animation they have been divided into several tabs:

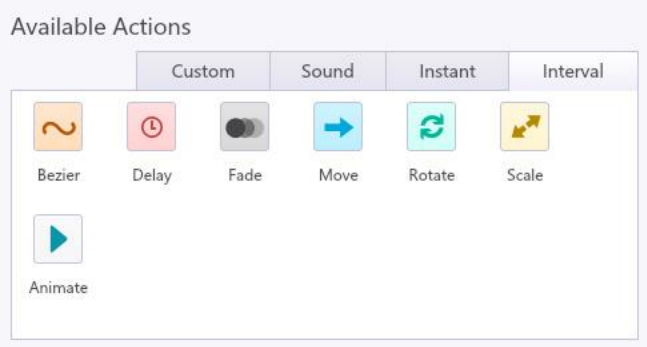

Figure 55 – Available Actions

- **Actions range**: where execution occurs gradually over the time interval that lasts.
- **Instant Actions**: those that occur instantaneously, at time zero.
- **Sound Actions**: also snapshots, but they have to do with sound and music themes of the project
- **Personalized Actions**: custom actions you can create from other actions. For more information go to *Advanced Concepts*.

TO and BY Modes

The majority of *Interval Actions* have a *Mode* property that can take values **TO** and **BY**. These values influence how the parameters are applied to the action on the *Actor* (or *scene*), so it is important to understand the difference between the two.

Let's use a *Move* action as an example to explain the difference between the two values. If we have a Actor in the position X = 500, Y = 300, when executing a *Move* action with a TO mode and values X = 200 and Y = 300, the actor will move **following values specified in the action** (gradually, during the duration of the action, like all interval actions); but if you executed the same *Move* action with a BY mode, and the same values X = 200 and Y = 300, the actor will **move to the position resulting from adding these values to its current position**, that is to say, X = 700, Y = 600.

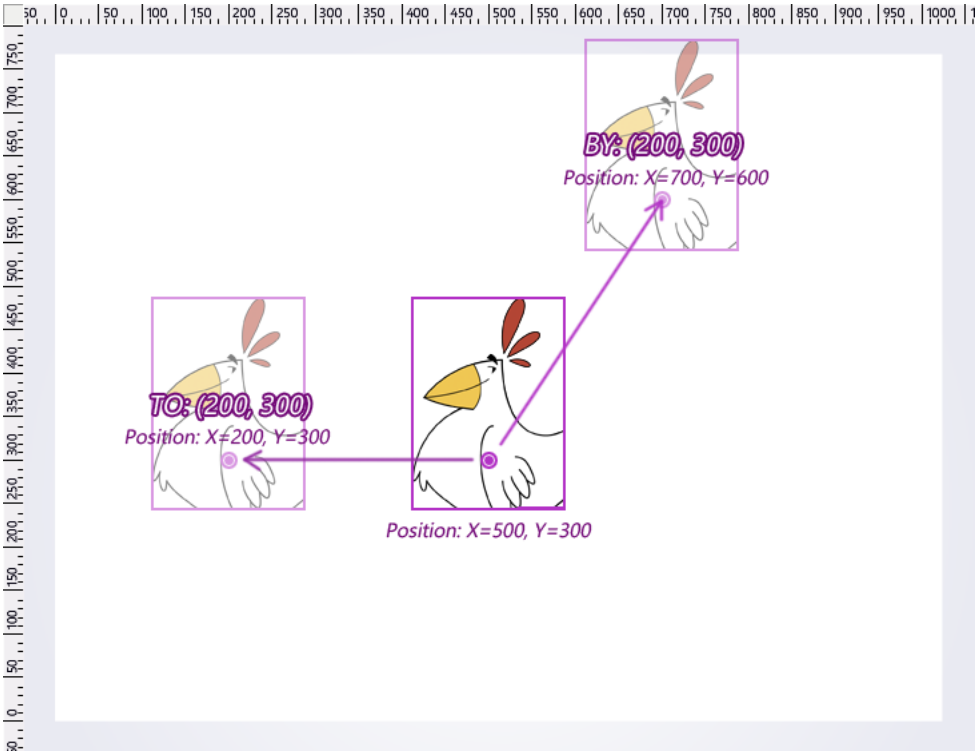

Figure 56 –TO and BY mode positioning example

Both modes will be useful depending on the situation. If for example you want an actor to move 100 pixels to the left every time you touch (*TOUCH*) it, we can (combine) use a *Move* action with BY mode and the values X = -100, Y = 0. Or, if you want the actor to be dropped (*DROP*) in the position X = 100, Y = 100, we will use a *Move* action and TO mode with the values X = 100 and X = 100.

NOTE: There is often confusion because if the actor performs an action with the *TO* mode with the same values entered before, it will appear that the action "has no effect". Actually yes it does, but the effect does not change its properties. Be careful with this!

Easing

The *Interval Actions* also include the ability have a **smoothing effect** on the execution of an action, in this way the animation does not seem homogeneous and achieves a more natural or fluid appearance.

The parameters that control the *easing* of the action are:

- **Smoothing Function:** indicates the easing function to apply to the action. You can take the following values:

- **EaseNone** (No easing): doesn't apply any smoothing effect to the action, and it will run evenly over its duration.

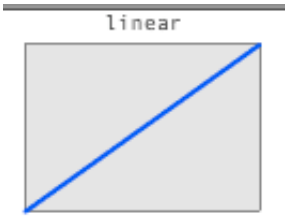

Figure 57 – Ease None function graphic

- **EaseExponential** (Exponential): a strong acceleration (deceleration) will be applied to the action.

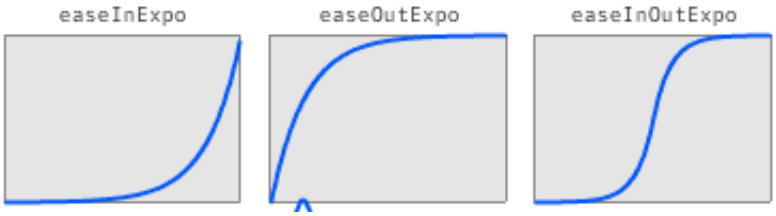

Figure 58 – Ease Exponential function graphic

- **EaseSine** (Sine): a soft acceleration (deceleration) will be applied to the action.

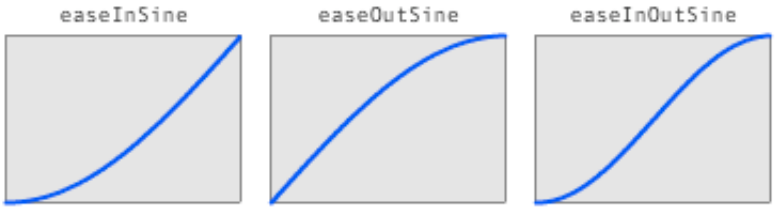

Figure 59 – Ease Sine function graphic

- **EaseElastic** (Elastic): la función de suavizado elástica provoca en la animación un efecto de vaivén simulando al de un elástico.

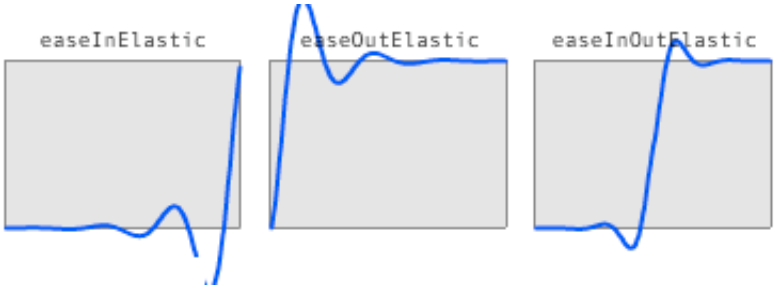

Figure 60 – Ease Elastic function graphic

- **EaseBounce** (Rebound): la función de rebote simula un efecto rebote al principio (final) de la ejecución de la acción.

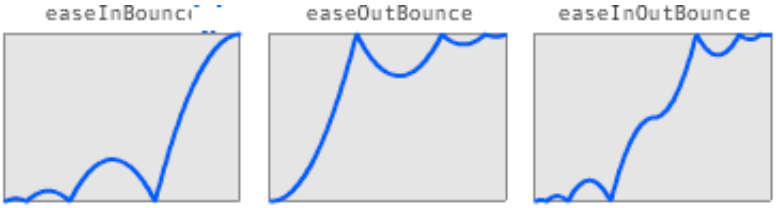

Figure 61 – Ease Bounce function graphic

- **EaseBack** (Backspace): la función de suavizado con retroceso simula una anticipación a la acción.

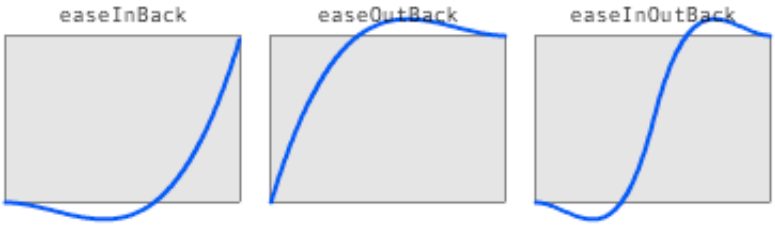

Figure 62 – Ease Back function graphic

- **Smoothing mode**: indicates how to apply the chosen smoothing function to the action.
  It features the following values:
  - **IN** (Input): the smoothing function is applied to the beginning of the action.
  - **OUT (**Output): the function is applied at the end of the action.
  - **IN-OUT** (Input-Output): the function is applied to both the beginning and end of the action.

Interval Actions

In this tab, the actions are grouped together and executed along a time interval. Most of these actions modify the graphical properties of the elements that trigger the *Animation* that it has, directly influencing their representation on screen.

IMPORTANT: If you set the *Duration* of the action to 0 milliseconds action, the action is executed instantaneously, so that the result will be applied immediately on the element that was triggered.

Figure 63 – Interval Actions

- **Bezier**: This action moves the element along a curve. The action is so named *Bezier* to describe the curved movement using a cubic Bezier curve. These curves are widely used in vector animation programs like *Adobe Flash*, or graphic design programs such as *Photoshop* or *Illustrator* (Pen tool).

  An example of cubic Bezier curve looks like this:

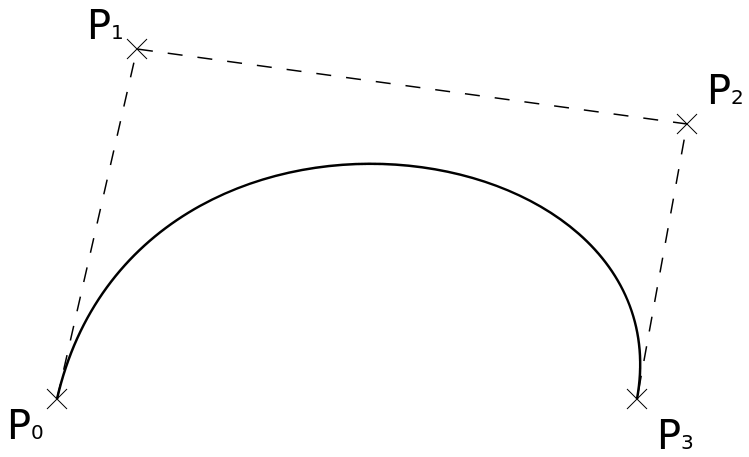

Figure 64 –Bezier Curve example

And it corresponds with the values of the action form as follows:

- **Point 0**: the position of the element at the time of the action is executed. This value can't be specified on the form.
- **Point 1**: the first checkpoint. Corresponds with **Control P. 1**.
- **Point 2**: the second checkpoint. Corresponds with **Control P. 2**.
- **Point 3**: the final position of the element after executing the action. Corresponds to the field *End Position*.

See the *Examples Section* of the *forum* for more examples of using Bezier action.

- **Delay**: This action has no effect. It merely serves to introduce a delay in the sequence to execute another action after it.
- **Fade**: this action can change the opacity of the element, so we can make it partially or fully transparent gradually, during the execution of the action.

> **NOTE**: Remember, the opacity of an element varies from 1 (completely opaque) to 0 (fully transparent).

Here the *Mode* of the action acquires different values than the rest of the actions located in the same tab, such as:

- **TO mode**: if you choose this mode, you can specify opacity value of the value at the beginning of the action (*Starting Alpha*) and what value it will have at the end of the action (*Final Alpha*).

    - **IN mode**: In this mode, the opacity of the element goes from 0 to 1. That is to say, the element "appears" in the scene.
    - **OUT mode**: in this mode, the opacity goes from 1 to 0, "disappearing" from the scene.
- **Move**: with the *Move* action, we can move to the element within the scene. The actor will move in the specified amount of time, from the position it was in at the time the action was triggered, to the position indicated in the *Position* field.
- **Rotate**: using this action can change the *Rotation* of the actor gradually over time. The value in the *Angle* field determines the quantity of the rotation, in degrees, from -360º to 360º. If you use positive values, the rotation will move in a clockwise direction, and negative values means the rotation will move counter-clockwise (opposite direction), we can see an example of this in the following graph:

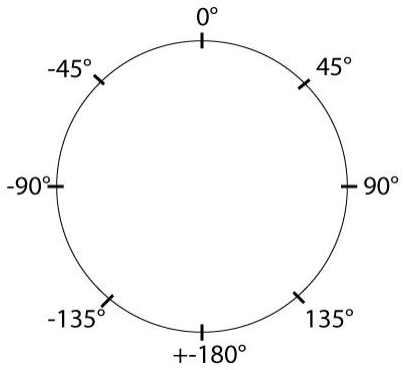

Figure 65 – Rotation diagram of an Actor

It is important to note that with the **TO Mode** when you indicate that you want your actor to rotate from one position to another **the actor or element will rotate in a way so that it arrives at the indicated value with the least amount of movement**. This breaks the positive/negative rule we mentioned earlier (but only with the *TO Mode*). So for example you choose to rotate from 90º to 45º, the animation will rotate counter-clockwise because it is a shorter distance to rotate counter-clockwise as opposed to making an entire clockwise rotation and arriving at 45º.

IMPORTANT: Remember that in the TO mode, to rotate 360 degrees is the same as rotating to -360º or 0º. If the rotation of the element was 0 ° at the time of the action began, the action "would have no effect."

- **Scale:** the *Scale* action allows us to change the scale, and therefore the size of the element in the scene. The scale can be set in the *Scale* field of the action form, both in width with parameter X, and height with parameter Y. These **parameters vary in a relative scale**, so that a

value of 1 indicates the original size of the element in this parameter, a value of 2 indicates twice, etc.

- **Animate**: This action allows us to **animate the element frame by frame**, i.e. it gives us the ability to animate the element in a traditional way, like a *cartoon*, changing your chart over time.

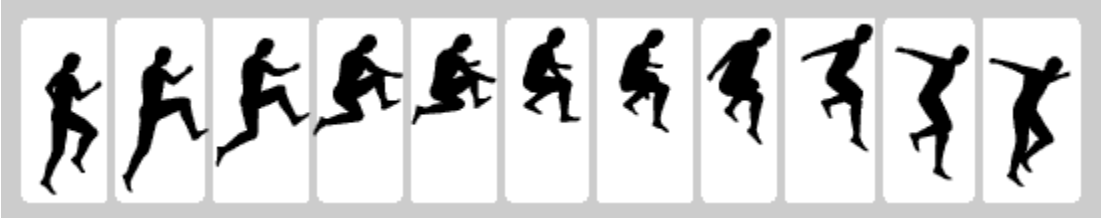

Figure 66 – Frame Animation example

To use this action, **you must first create or add a Spritesheet** including the frames of animation.

Once it's been created, select the *Spritesheet* field and then press the *Edit frames* button to select the frames that make up the animation.

Instant Actions

This is where the actions that are executed instantaneously are grouped (even the actions whose effects continue for a period of time after their initial execution). He were can see the different type of *Instant Action* events:

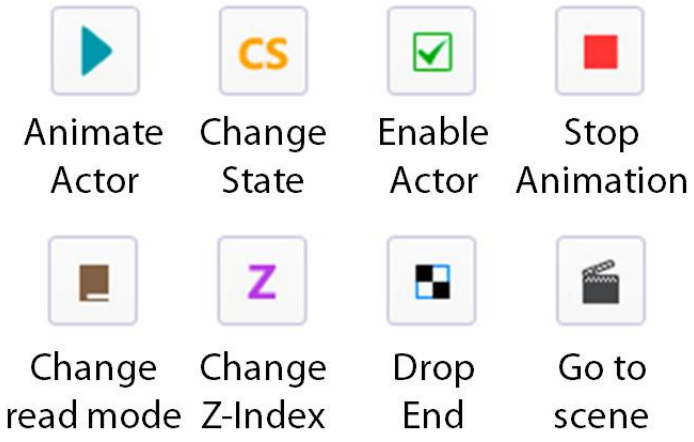

Figure 67 – Instant Actions

- **Launch event**: at the beginning of this section, we commented that the animations are activated only when the element that contains the corresponding *Even*t is triggered. Thus, one could run an animation from user interactions (*TOUCH, DRAG,* etc.) or after receiving automatic system events (*ENTER, PAUSE,* etc.). However, thanks to the *Launch event* we can "fool" the elements into believing their event has been triggered and thus reproduce the animation associated with it.

- The most common situation where this action is typically used is when you want to TOUCH an element of the scene and have another element, whose event has not been triggered, execute its animation as well. Such as pressing a button that opens a door.

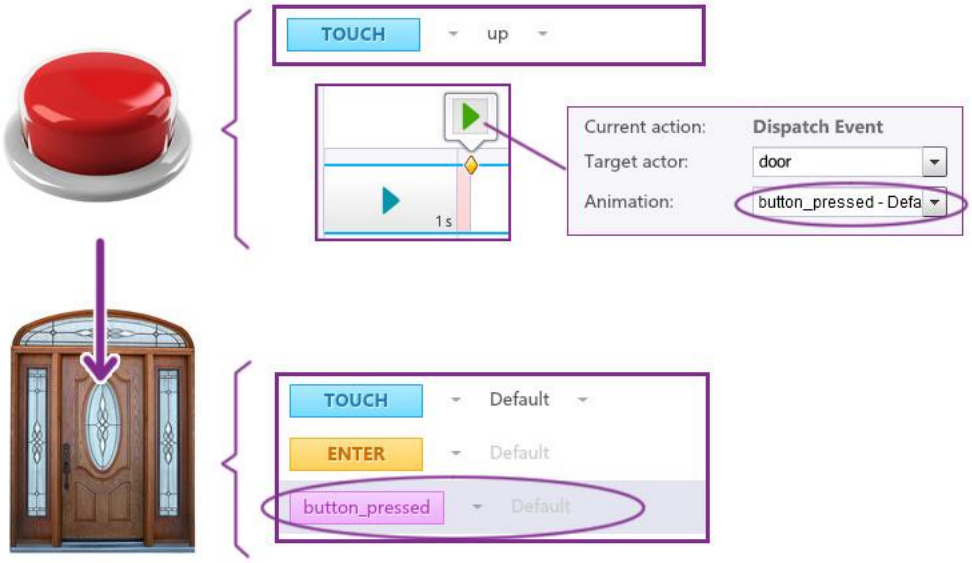

Figure 68 – Dispatch Event example

Normally (especially when you want an element to execute a behavior only when another element's animation is executed) the element objective of the Launch Event action is to trigger the **Custom Event** action too, meaning that neither one corresponds to the definitions of the system (TOUCH, ENTER, DRAG, etc.).

For more information about using **Custom Events**, see *Animation Events*.

- **Status Change**: This action is used to change the status of any element of the scene, so we can implement the ***Animation* State**.

Animation State is a very powerful PlayTales ® Builder ™ tool. Thanks this features we can associate different behaviors with an element when one *Event* is triggered, making our story more vibrant and lifelike. For more information on *Animation States*, go to *Animation States*.

- **Enable / Disable actor**: if we want to determine that the Event of a scene will not be triggered, and therefore no behavior will be executed, we'll use the *Enable/Disable actor*. This action is especially used if, for example, we want to make an actor untouchable until the user has touched another actor beforehand, like if we wanted the user to touch a door to open it, and then touch the actor so that it enters through the opened door.

- **Stop animation**: With this action, we can stop an animation that was activated at this moment. To properly use this action, we must bear in mind that:
    - If an element is going to reproduce an animation that is already activated, because it's *Event* has been triggered and activated, the animation is not interrupted or restarted, i.e. nothing happens.
    - If a *Stop Animation* action is used and the animation stops halfway, the changes that have been made in the element cannot be undone. The element will retain the modified values that it had at the moment of the stop.
    - If the same animation is played again, the actor does not retrieve the values that it initially had, if not taking as initial values which had to stop the animation. This is especially important when using **BY Mode** actions.
    - If you try to stop an animation that has not been activated, nothing happens. Just to be safe, use the *Stop Animation* action and this way you won't end up producing strange or unwanted effects.

NOTE: If you want to **execute an animation element "from scratch" every time its *Event* is triggered and activated**, execute a previous "initialization", so that 1) the *Animation* stops, 2) the values are restarted, and 3) the *Launch* of the animation objective is renewed. You can see an example of how to do this in the *Examples* section of the *forum*.

- **Go to scene**: normally, to move from one scene to another you need to push the buttons located on the bottom of the screen. Thanks to this action we can directly jump to any scene of our book and break the normal flow of reading scenes.

NOTE: With this action "secret scenes" can be defined, which cannot be accessed otherwise than by interacting with a specific item, and also stories such as "Choose your adventure", which gives the user the option to follow a story by their own choosing.

IMPORTANT: The *Go to Cover (Cover Page)* works as the **PlayTales ® Default Cover Page** defined within the project *Options*. If your project does not include a default cover page, do not use this value.

- **Change Reading Mode**: PlayTales ® Builder ™ implements several **Reading Modes** to give the user the possibility to listen to the narration (*Read to me*), the user can choose to read and interact with the book alone (R*ead it myself*), or play the book automatically (*Autoplay*).
  If your project includes a *Default Cover Page*, the Reading Mode can be selected from there depending on the book configuration you've set up. You can create and personalize the *Cover Page Menu* and the user can change the reading mode at any time, even during the story.

- **Change Z Index**: the elements of a scene are positioned on an X and Y axis according to the X and Y coordinates established within their properties menu and through the Move action. But to define what elements "appear" in front of others, the *Z Index* coordinates are used. The *Move* action does not allow you to change the Z coordinates of the elements, if not it can be changed through the *Change Z Index* action.

  Thanks to this action, we can make the elements dynamically change their order of appearance in the scene, positioning them in front of and behind other elements while executing animations.

  NOTE: Remember that the higher the Z coordinate number is the more "forward" it will appear in front of other elements in the scene who have lower Z coordinates.

- **Drop End** (available only in animations with ***DROP Event***): if we want a *Draggable* element (i.e., you can drag around the screen) to execute a specific behavior when it is dropped or released (*DROP*) on a particular item, we should use the *End Drop* action.

  This action lets you define which events trigger **the element itself includes the action**, when the element is dropped onto a specific item. For example, in the following case, for the *T-Shirt* actor combines a *Drop-End* action animation with a *Drop Event*:

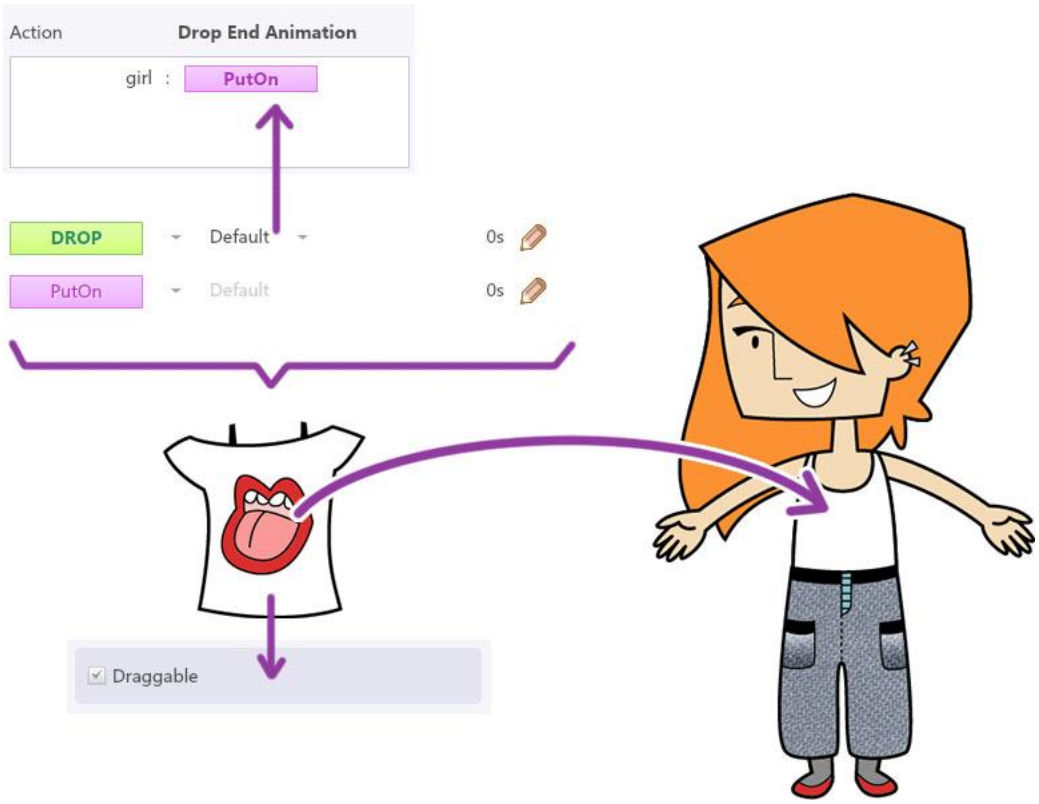

Figure 69 - *Drop-End* example

We are saying that the *T-Shirt* actor will trigger a custom *Put On* event when this actor is dropped on top of *Girl* actor, as well as trigger the custom event *Keep* when a DROP is made on top of the *Wardrobe* actor.

You can get the example in the following direction: http://playtalesbuilder.com/resources

NOTE: To find out if an element has "collided" with another, we can use the **Bounding Box** as a reference. Consult *Create an Actor and edit its properties* for more information.

To create new *Element – Event* pairs, you must select the corresponding actor and event in the combobox, and press the **(+) button**. You can delete a pair at any time with the **delete button**.

IMPORTANT: Only send the element when the *Drop End* is executed that is associated with **the first element with the "collision"** (the one with the largest Z coordinate). That is, if there are two elements which can collide with one above the other, and collides with the top one, the element below is ignored. In addition, all elements are ignored in the scene that are not registered in the table.

Sound Actions

In this tab you will find that actions can be executed instantly as well, but it has to do with playing sounds in the engine. To correctly utilize the sound actions, we recommend looking at the *Advanced Concepts* section. Within Sound Actions the following actions are available:

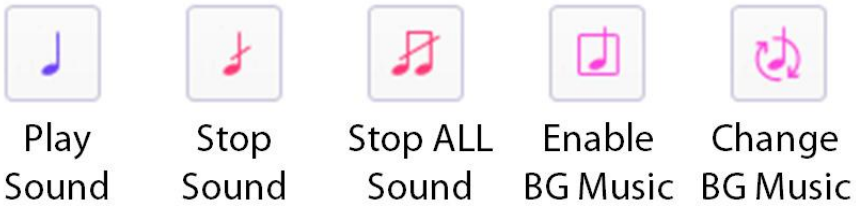

Figure 70 – Sound Actions

- **Play Sound:** With this action we can play sound file we've previously added to our *Library*. This action should only be used to play **sound effects** (SFX) and **narrations** (Voice). If you want to play **background music**, use the *Scene* or *Project* properties, or the actions related to playing background music.

> **IMPORTANT**: As the memory of tablets and phones is reduced, and being that sound resources consume a lot of memory, we recommend you optimize the audio files and use sound actions correctly.

- **Stop Sound**: Use this action when you want to stop a sound that was playing at this time. Just like with *Play sound*, this action should only be used for *sound effects* and *narrations*. To stop *background music*, use the action *Enable / Disable background music*.
- **Stop all sounds**: If at any time you want to stop all *sound effects* and *narrations* of a scene, rather than adding a *Stop Sound* action for each sound you want to stop, you can use this action to do so more easily/quickly.
- **Enable / Disable background music**: When there is *background music* you want to stop playing (or replay) at a particular moment in the scene, use this action instead of the equivalent for *sound effects* and *narration*: *Stop Sound*.
- **Change background music**: Each scene has a property in its properties form to select what background music you want to play at the beginning of the scene, but if you want to play different music in the middle of the scene you can use this action to change it at any time.

Custom Actions

In our book if there is an animation or behavior that you'd like to place several times throughout the project scenes, an animation that is very complex and difficult to build, it is very inconvenient and time-consuming if you have to rebuild it every time you want to use it. This is also the case if you need to copy and paste animation over and over again, or if you need to tweak something after you've already pasted it in several different scenes. That is why PlayTales ® Builder ™ gives you the ability **to define new actions as part of the default actions that are available in the program**. These actions are called ***Custom Actions.***

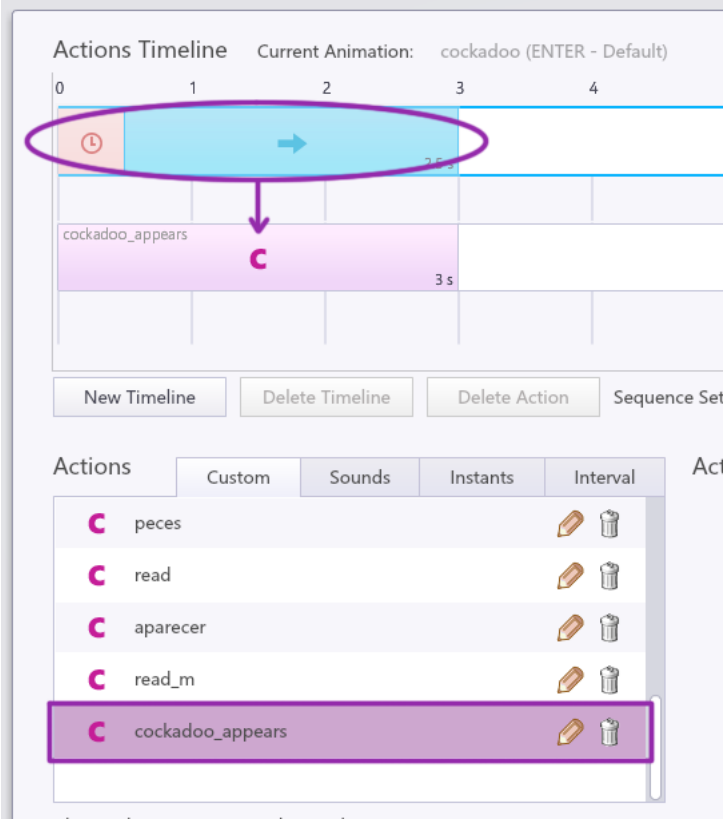

Figure 71 – *Custom Action* example

In this last tab we can find all the *custom actions* that we have created in our project.

To **create a *custom action*** from a parallel block of action sequences (which make up the animation), we only need to press the **Convert to Custom** button, choose a identification name for the new action, and it will be available within the *custom actions* tab in the project. The new *custom action* will group all the components that make up this animation.

Figure 72 – Create *Custom Action*

NOTE: You can edit the *custom action* at any time by double clicking on the action in the sequence, or by pressing the pencil icon from the list.

Custom actions allow us to save time in developing and optimizing the memory of our project on the device, but there are some important restrictions to keep in mind:

- **Concrete animations cannot be changed**: a custom action cannot be changed in some parameters for a specific case. This is common for all the animations used within the concrete animation, and if modified these changes will affect the concrete animation wherever it is used within the project.
- **You can't include cycles or repetitions**: if you want your *custom actions to loop or repeat*, in order to not produce inconsistencies, create a *custom action* sequence and loop the animation within it.
- **Don't use *custom actions* within other *custom actions*:** as it can lead to infinite loops and faults within in the project.

Create an animation

Once you understand how the PlayTales ® animations function, creating animations can be as simple as pressing the **(+) button** on the *Animations* tab of the actor or the scene, or via the menu in *Animations*> *Create animation*.

A new animation is created in empty list item animations, with the event and the state still to be allocated.

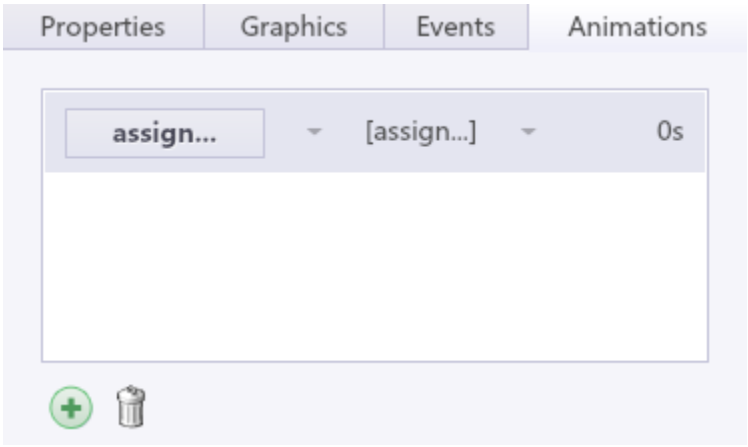

Figure 73 – New Animation

Before editing the animation, it's necessary to assign the corresponding *Event* and *State* its activation.

Edit an animation

After creating an animation and assigning your *Event* and *State*, for editing, just click the pencil icon, double-click the animation, or click the *Animations* > *Edit animation* menu. The animation edit menu will then open and you can edit the animation.

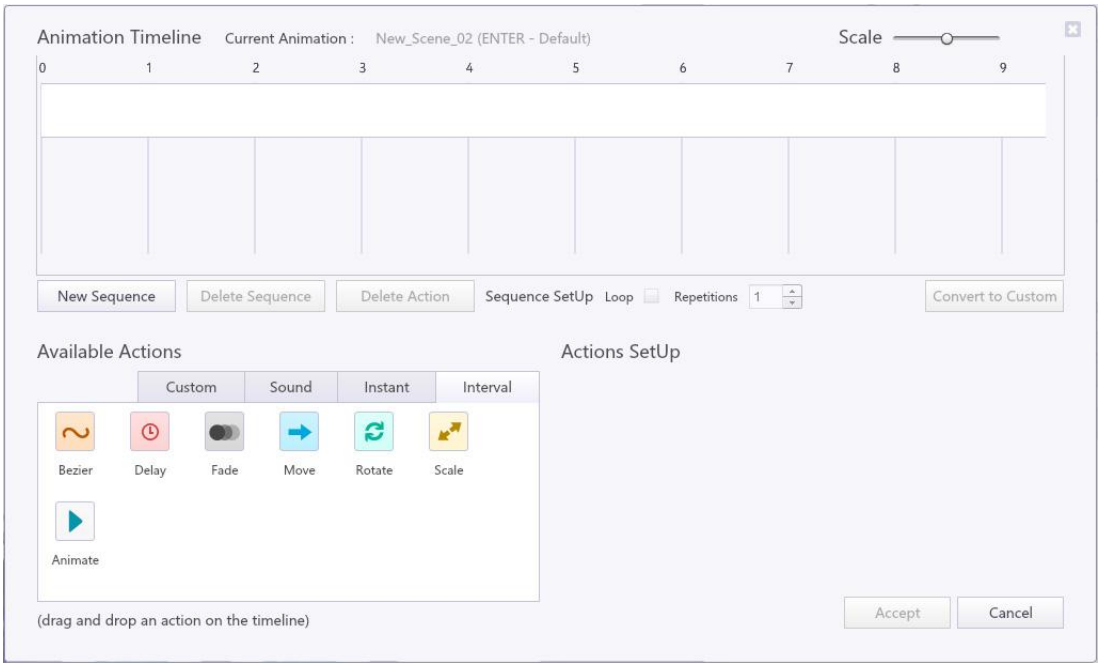

Figure 74 – Animation Edition Panel

From this menu you can configure all parameters of the animation: add and configure the *values of the repetition of sequences*, add actions to sequences by dragging them from the list of *available actions*, create *custom actions*, etc.

Test your project on a real device

We're almost finished with our project! But…how does it look on a real mobile device?

In order to see the results of the scenes we've made using the PlayTales Builder, we can add the animations and elements of the scenes to the **Simulato**r, which is included in the program. Open the Simulator and follow the steps in the *Launch PlayTales Simulator* section to see how it's done.

It is very important to test the project on a real device before it is published and sold. We need to see how the project will appear on a phone or tablet and know what the user's experience will be like. You should **deploy** the project onto a device using the intuitive and helpful tools available in PlayTales® Builder™.

Figure 75 – *Con los transportes aprendo*, de Dada Company

Deploy the project from PlayTales® Builder™

To launch our project from PlayTales ® Builder ™ to our device, we must select in the menu File> Deploy project, if we visualize the entire project as it would already published, or File> Expand current scene, to test only the scene on which we are working. This last option is useful for rapid deployment, since resources are only needed in the current scene.

To launch our project from PlayTales ® Builder ™ onto a device, we should select the menu option *File> Deploy project*, if we want to see the entire project just as it would appear once its published. Another option is to select *File>Deploy current scene*, which will deploy the current scene you are working on. This last option can be used for a quick deploy, since you'll only be using resources from the current scene.

These are the deployment options depending on your device:

- **Deploy iOS**: for iPhone, iPod, iPad, and generally any Apple devices
- **Deploy Android SD**: if we want to deploy the project onto an Android device that only accepts projects featuring standard definition (SD).
- **Deploy HD Android**: to deploy high definition (HD) projects onto an Android device.

NOTE: For Android, PlayTales® Builder™ Viewer, the program which receives the project in the device, will tell you which kind of projects supports your device.

Selecting any of these options will open the *project deployment wizard* which will immediately begin to prepare resources for deployment by analyzing errors and optimizing elements for deployment onto the device.

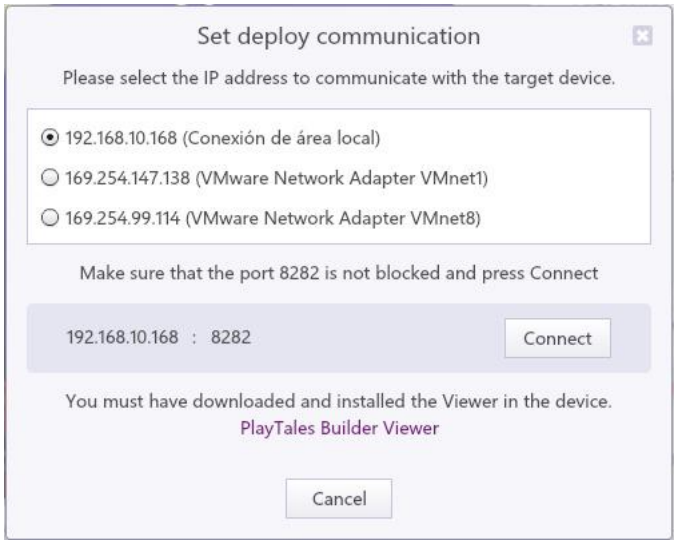
Set deploy communication
Please select the IP address to communicate with the target device.
192.168.10.168 (Conexión de área local)
169.254.147.138 (VMware Network Adapter VMnet1)
169.254.99.114 (VMware Network Adapter VMnet8)
Make sure that the port 8282 is not blocked and press Connect
192.168.10.168 : 8282
Connect
You must have downloaded and installed the Viewer in the device.
PlayTales Builder Viewer
Cancel

Figure 76 – Deploy Project wizard

For Builder to communicate with the device internet connection is required. If your computer does not have internet access at this point in time a message will appear notifying you fix your connection.

Once all the resources are prepared, the wizard will ask you to select the IP address to use for the transfer. Select one of the available IP options listed (usually there is only one).

IMPORTANT: In order to establish the connection, both the Builder and the device must be connected to the same Wi-Fi network.

After selecting the IP address, if you click ***Connect***, the Builder will begin transmitting the project information to the destination indicated in the window, and wait for the device you connect.

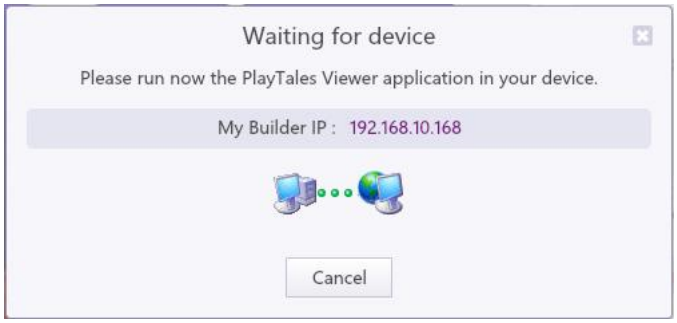

Figure 77 – Waiting for device

IMPORTANT: Make sure there are no antivirus or firewalls activated that may block the network port 8282, the network that Builder uses to communicate with the device. Also make sure the port is not in use on that network.

Receive project in PlayTales® Builder™ Viewer

To view the project on the device, it is necessary to install and run the PlayTales ® Builder ™ Viewer application. This application is free and can be downloaded from the official app store in your device, such as Apple's App Store or Android's Google Play.

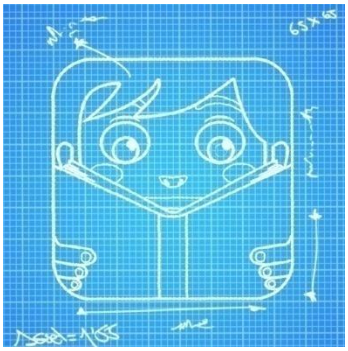

Figure 78 - PlayTales® Builder™ Viewer icon

- **PlayTales® Builder™ Viewer in iTunes:** https://itunes.apple.com/es/app/playtales-builder-viewer/id523576672
- **PlayTales® Builder™ Viewer in Google Play:** https://play.google.com/store/apps/details?id=com.gi.playtales.builder.viewer

When you run the application, you will see a menu featuring several options. You can either select Connect, My Projects, or About; to connect your device with the Builder click on the **Connect** option.

Figure 79 - PlayTales® Builder™ Viewer in iPhone

As soon as you hit Connect, PlayTales ® Builder ™ Viewer starts searching for "Builders" that are connected to the same Wi-Fi network. When it finds one, it will show a list wherein you can select one to transmit the project to.

Deploy a project from a Mac

In order to correctly convert files in an optimized format necessary for different systems (for example, AAC for iOS sounds, or rescaling graphics for *Android* phones) PlayTales® Builder™ uses **tools that must be manually installed on Mac computers**.

NOTE: For Windows, these tools are included with the installer and are installed automatically. Therefore, you do not need to take the following steps described in this section.

Specifically, the following tools are used:

- **FFmpeg** (http://www.ffmpeg.org):

  FFmpeg is a multiplatform tool that offers a solution to record and convert files and audio and video streams. It is used to convert the format of sound files used by Builder for different devices. It is necessary if you want to listen to sounds on a device.

- **ImageMagick** (http://www.imagemagick.org):

  ImageMagick is a tool to create, edit, compose, and convert images to bitmap. In Builder it is used to optimize the rescaling of graphics to be displayed on phones. Installing this is optional. It is only needed if you want to experience higher-quality graphics on your phone.

These tools are used automatically in Builder, so you don't need to know how to use them or how they work. You only need to install them.

This installation can be done in various ways, but the easiest way is to install them through installation managers such as MacPorts (http://www.macports.org/) or Homebrew (http://mxcl.github.com/homebrew/).

Below, we will demonstrate step by step **how to install the applications using Homebrew.**

Install Homebrew

To install Homebrew on your system as shown on the official website (http://mxcl.github.com/homebrew/):

1. Open the **Terminal** (*Applications > Terminal*).
2. Enter the following line and press Enter:

```
ruby -e "$(curl -fsSL https://raw.github.com/mxcl/homebrew/go)"
```

3. Press Enter to install and then type in your user password. The program will start downloading necessary files and will install automatically.
4. Once installed, you may need additional libraries and components for Homebrew to run properly. Type the following line:

```
brew doctor
```

   Follow the program instructions. It is possible that you may have to:

   - Install or update **Xcode** (*Free of charge through the Mac App Store*): http://www.apple.com/es/osx/apps/app-store.html
   - Install the package, **Command Line Tools** in Xcode.
   - Install or update **Git** (*Free of charge through the owner of Homebrew*):

```
brew install git
```

5. When you have completed everything, if we execute `brew doctor`, Homebrew will tell us that everything is now ready to use.

Install FFmpeg

To install FFmpeg with the additional *libvorbis* library in order to convert sounds from MP3 to OGG (Android).

6. Type the following line into the terminal and press Enter:

```
brew install ffmpeg --with-libvorbis
```

7. Type in your user password if necessary. The program will download automatically and install the necessities to your system.

Install ImageMagick

To install *ImageMagick*:

8. Type the following line into the terminal and press Enter:

```
brew install imagemagick
```

9. Type in your user password if necessary. The program will download and install everything you need to run the program.

Now you have everything you need to deploy your projects from your Mac without quickly and easily! To be sure that both programs installed correctly, through the terminal you can run `which ffmpeg` and `which convert`. If you installed the programs with *Hombrew*, both should be found in `/usr/local/bin`.

If you have any problems during the installation of the software or while launching the project, please contact Builder´s technical services at info@playtalesbuilder.com.

Publish your project

You're finally finished with your book! You have created a well developed story featuring different scenes, distinctive characters, interesting and entertaining interactive elements. Now that you've tested the project on a real device using the simulator, the only thing left to do is publish your book so the whole world can enjoy it!

Figure 80 - *How Little Plane Learned to Fly*, by Ekaterina Syromyatnikova

Post your project from PlayTales® Builder™

The publication of the project is done from within the Builder itself and is the last step in the creation of the project. Re-publishing a project is not recommended, **you should** thoroughly test the book using the simulator as well as make sure it meets the high-quality standards set for PlayTales books ® before publishing it.

To publish the project, open the *publishing wizard* by clicking the *File* > *Publish Project* menu. This option is enabled only if the project has at least one scene and is saved to disk.

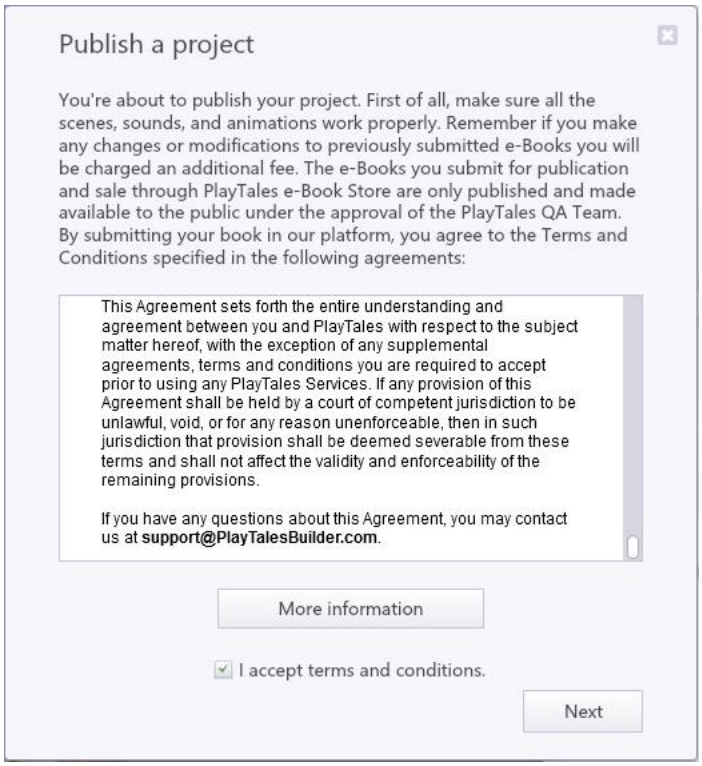
Publish a project
You're about to publish your project. First of all, make sure all the scenes, sounds, and animations work properly. Remember if you make any changes or modifications to previously submitted e-Books you will be charged an additional fee. The e-Books you submit for publication and sale through PlayTales e-Book Store are only published and made available to the public under the approval of the PlayTales QA Team. By submitting your book in our platform, you agree to the Terms and Conditions specified in the following agreements:
This Agreement sets forth the entire understanding and agreement between you and PlayTales with respect to the subject matter hereof, with the exception of any supplemental agreements, terms and conditions you are required to accept prior to using any PlayTales Services. If any provision of this Agreement shall be held by a court of competent jurisdiction to be unlawful, void, or for any reason unenforceable, then in such jurisdiction that provision shall be deemed severable from these terms and shall not affect the validity and enforceability of the remaining provisions.
If you have any questions about this Agreement, you may contact us at support@PlayTalesBuilder.com.
More information
I accept terms and conditions.
Next

Figure 81 – Publish a project

The first screen of the *publishing wizard* lists the terms and conditions that you must accept before continuing. Please read them carefully, and if you agree with them, check the box and click *Next*.

At this point, if we were not identified in the application, the wizard prompts you to enter your user name and password. If you are already logged in, the wizard will skip this step. If you aren't registered with PlayTales® Builder™ yet, you can do so via the official website: http://www.playtalesbuilder.com

Finally, before showing the publication process, the wizard will show a confirmation message, showing some additional information. If you press *Next* now, the process will start.

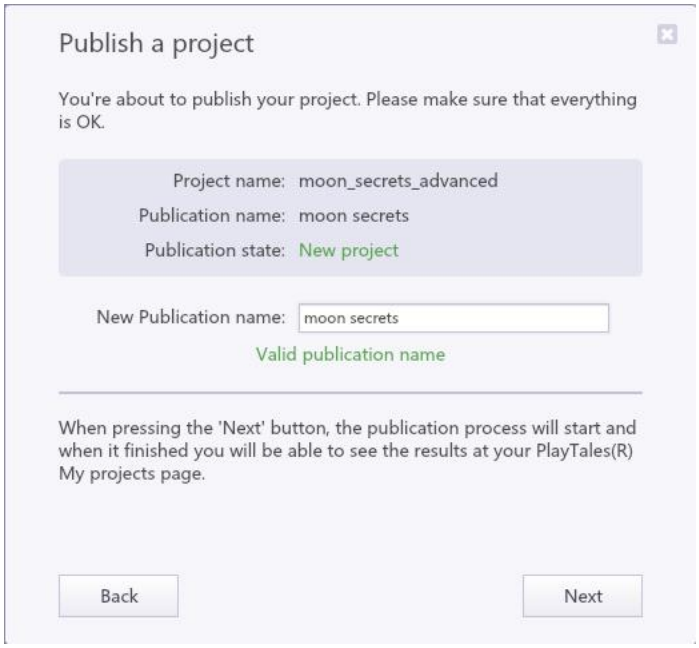
Publish a project
You're about to publish your project. Please make sure that everything is OK.
Project name: moon_secrets_advanced
Publication name: moon secrets
Publication state: New project
New Publication name: moon secrets
Valid publication name
When pressing the 'Next' button, the publication process will start and when it finished you will be able to see the results at your PlayTales(R) My projects page.
Back
Next

Figure 82 – Publish Project confirmation menu

NOTE: If the project has previously been created and published and has been submitted for *quality assurance* (*QA*), it cannot be published until the QA process is complete.

Advanced Concepts

This section will present a number of different concepts that you can use to take advantage of Builder's tools.

This section is currently under construction! See *advanced examples* in the **official PlayTales® Builder™ forum** to see various explanations and concept examples.

http://www.playtalesbuilder.com/forum/10/

Sorry for the inconvenience!